Tossing Coins with an NP-Machine

Edgar Daylight *

Posted Date: 29 July 2025

doi: 10.20944/preprints202507.2367.v1

Keywords: nondeterminism; correlated coin tosses; P; NP

 doi:10.20944/preprints202507.2367.v1

Article

Tossing Coins with an $\mathcal{NP}$-Machine

Edgar Graham Daylight

a.k.a. Karel Van Oudheusden, Department of Computer Science, KU Leuven, Celestijnenlaan 200a, Box 2402, 3001 Leuven, Belgium; egdaylight@dijkstrascry.com

Abstract

In computational complexity, a tableau represents a hypothetical accepting computation path p of a nondeterministic polynomial time Turing machine N on an input w. The tableau is encoded by the propositional logic formula ψ, defined as $\psi = \psi_{cell} \wedge \psi_{rest}$. The component ψ_{cell} enforces the constraint that each cell in the tableau contains exactly one symbol, while ψ_{rest} incorporates constraints governing the step-by-step behavior of N on w. In recent work, we reformulated a critical part of ψ_{rest} as a compact Horn formula. In other work, we evaluated the cost of this reformulation, though our estimates were intentionally conservative. In this article, we provide a more rigorous analysis and derive a tighter upper bound on two enhanced variants of our original *Filling Holes with Backtracking* algorithm: the refined (`rFHB`) and the streamlined (`sFHB`) versions, each tasked with solving `3-SAT`.

Keywords: nondeterminism; correlated coin tosses; $\mathcal{P}$; $\mathcal{NP}$

1. Introduction

Let N be a nondeterministic Turing machine (TM) that, on input w of length n, either accepts or rejects w within n^k steps for some constant k. A conservative, deterministic simulation of N requires up to 2^{n^k} steps, with each computation path corresponding to a chronology of binary nondeterministic choices.

In prior work [1], we conveniently assumed that N's stepwise behavior could be concisely captured by a Horn formula ψ^{η}_{step}. Later, we confirmed this conjecture in [2]. These findings now allow us to concretely move beyond the classical top-down chronology of N's computation toward a non-sequential understanding of computability.

Traditionally, an accepting path of N on w can be represented by an accepting tableau—a two-dimensional matrix of cells—and formalized in propositional logic as a satisfiable 3cnf-formula ψ. In our approach, however, we relax the tableau structure by allowing "holes," replacing the 3cnf-formula ψ with a Horn formula ψ_{trim} of size $O(n^{\kappa})$ for some constant κ. (Determining the satisfiability of Horn formulas is, to date, substantially more efficient than for genuine 3cnf-formulas [3].) While ψ_{trim} is guaranteed to be satisfiable whenever ψ is, the converse does not necessarily hold.

Specifically, we define:

$$\psi_{trim} \;=\; \psi^{\eta}_{step} \wedge \psi_{start} \wedge \psi_{accept} \wedge \psi'_{cell} \wedge \psi^{1}_{extra} \wedge \psi^{2}_{extra}, \tag{1}$$

where each conjunct is a succinct Horn formula:

1. ψ^{η}_{step} captures the step-by-step behavior of N on w—with the Greek letter eta (η), resembling the Latin letter h—highlighting that the formula is a *H*orn formula.
2. ψ_{start} ensures that the initial row of the tableau encodes N's start configuration on w.
3. ψ_{accept} enforces that no cell in the tableau contains the reject state symbol q_{reject}.
4. ψ'_{cell} ensures that at most one variable is "turned on" per cell in the tableau, where a "hole" in the tableau refers to a cell where all variables are "turned off".
5. ψ^{1}_{extra} captures the spatial dynamics of the TM's head within the tableau (Theorem 1).

 doi:10.20944/preprints202507.2367.v1

6. ψ^2_{extra} expresses the inter-cell dependencies across distant rows (Theorem 2).

Crucially, if ψ_{trim} is satisfiable with the corresponding tableau containing no holes, then the original formula ψ is satisfiable too, implying that N accepts the input w.

How do the technical desiderata (1–6) relate to our prior work and to this paper? The formal definition of ${}^*\psi^{\eta}_{step}$, which appears as the first conjunct in the following definition of ψ^{η}_{step},

$$\psi^{\eta}_{step} \;=\; {}^*\psi^{\eta}_{step} \;\wedge\; \psi^{\mathtt{det}}_{step}, \tag{2}$$

is detailed in [2]. Extensive commentary of both conjuncts in (2) is provided in the present paper. (Notably, the second conjunct, $\psi^{\mathtt{det}}_{step}$, was not required in [2] due to the assumed presence of ψ_{cell} rather than ψ'_{cell}, thereby enforcing each cell in the tableau to contain precisely one "turned on" variable.)

The definitions of components 2-5 appear in [1]. Theorems 1 and 2—related to items 5 and 6, respectively—also appear in [1]. While we shall define ψ^1_{extra} and discuss Theorem 1 in this paper, our primary focus is on revisiting Theorem 2 and unpacking item 6, i.e., the "inter-cell dependencies across distant rows" that arise during the computation of N on input w. While the definition of ψ^2_{extra} is outlined in [1] for a `3-SAT` solver N^*, we provide a full, formal definition in this paper.

In essence, we offer a self-contained exposition of formula (1). This paper serves as a natural continuation of our previous works [1,2], while not subsuming them. Although certain formal definitions and proofs are deferred to those references, the present discussion remains accessible independently.

1.1. Methodology

Our approach centers on the formula ψ_{trim} and a timeless tableau—a matrix with $O(n^k) \times O(n^k)$ cells. When certifying the satisfiability of ψ_{trim}, this tableau typically contains holes, thereby encoding an exponential number of paths.

A simplistic reliance on a timeless tableau, where each cell's content is guessed independently, leads to a blow-up in deterministic time complexity—from 2^{n^k} to $2^{n^{2k}}$. To counteract this inefficiency, Theorems 1 and 2 from [1] convey two techniques that significantly reduce the number of required guesses.

- **Theorem 1: Compression via geometric constraints.** By leveraging a compression result that captures the spatial dynamics of the TM's head within the tableau, we shrink the search space for nondeterministic guesses. As a result, the deterministic time complexity is restored to the classical 2^{n^k} bound.

  Consider an initially empty tableau. By designating two specific cells—c^* (situated above) and c_* (located several rows below and, say, to the far right)—as head positions, we constrain the machine's behavior so that one or more binary nondeterministic choices collapse into deterministic transitions. In contrast, if only one cell holds a state symbol or if two nearby cells are populated with state symbols, the tableau exhibits a broader range of nondeterministic evolutions. For instance, if only c^* holds a state symbol, the machine may move freely left or right. However, if c_*, located far below and (say) to the right, also contains a state symbol, then some of the prior movements become restricted—only rightward transitions from c^* to c_* remain viable. This marks a shift from local stepwise reasoning to a more global, geometric form of constraint.

- **Theorem 2: Correlation in nondeterministic choices.** A second form of compression arises by distinguishing between a pure coin-tossing machine and a `3-SAT` solver modeled as a nondeterministic polynomial time TM N^*. Unlike the coin-tossing machine that produces independent bits, N^* generates correlated bits. This correlation allows for further compression of the tableau's nondeterministic behavior.

  For example, if the state symbol in cell c^* indicates that the machine has just produced a second coin toss of 1 (in one of multiple ways) and is about to toss a third coin, this

 doi:10.20944/preprints202507.2367.v1

constrains the allowable state symbol in a later cell c_* further down in the tableau. Due to inter-cell dependencies—embedded in the tableau's deterministic substructure—the machine may be forced to toss the next coin to 1. These subtle, long-range interactions are crucial for compressing the overall nondeterministic behavior. Importantly, the computation of N^* on input w does not unfold top-down but grows in an interleaved fashion across the rows of the tableau.

Building on the first theorem, Theorem 2 establishes that if the original formula ψ is satisfiable, then it can be satisfied within $K^{f(n,k)}$ steps with our ψ_{trim}-based approach, where $f(n,k) \approx n^{\frac{2}{3}k}$ and K is a constant. In this paper, we revisit Theorem 2 with more rigor and improved theoretical bounds.

Reflecting on Fortnow's recent speculation about compression [4], we argue that coin tossing by some nondeterministic polynomial time TM N—even when tasked to solve a cryptographic problem—is not entirely random after all. To our knowledge, Fortnow's exploration provides an early, if not unprecedented, framing of this topic. While the present author is familiar with the literature on Kolmogorov complexity [5], including a prior contribution to the field [6], no compelling connection is presently discernible between that work and the arguments advanced in this paper.

1.2. Task

The standard textbook formulation of an accepting tableau is given by:

$$\psi \;=\; \psi_{step} \wedge \psi_{start} \wedge \psi_{accept} \wedge \psi_{cell}, \tag{3}$$

where both ψ_{step} and ψ_{cell} are traditionally regarded as genuine 3cnf-formulas. Recently, however, we succeeded in re-expressing ψ_{step} as ${}^*\psi^{\eta}_{step}$, a compact Horn formula [2].

This improvement leaves ψ_{cell} as the sole non-Horn component, which ensures that each cell in the tableau contains exactly one variable that is "turned on." To move toward a purely Horn-based formulation, we weaken ψ_{cell} to a Horn formula ψ'_{cell} and compensate by introducing three additional Horn formulas: $\psi^{\mathtt{det}}_{step}$, ψ^1_{extra}, and ψ^2_{extra}. Together, these components yield the Horn formula ψ_{trim}, as defined via formulas (1) and (2).

Remark 1. *A Boolean formula in conjunctive normal form (cnf) is referred to as a* **3cnf-formula** *if each clause consists of exactly three literals. A cnf-formula is referred to as a* **Horn formula** *if every clause contains at most one positive literal. Standard definitions appear in Appendix A.*

The task laid out in this paper is to formalize each newly introduced conjunct in formula (1), to demonstrate that all components cohere and function in unison, and to establish a tight upper bound on the running time of our two ψ_{trim}-based algorithms: `rFHB` and `sFHB`. These represent refined and streamlined variants, respectively, of the original `FHB` algorithm. All three algorithms employ a standard `HORNSAT` solver $\mathcal{H}$ and incorporate external user actions, automating her interventions while embedding backtracking as an intrinsic capability.

In revisiting our initial cost analysis of `rFHB`, we note that Theorem 2 of [1] was proven under an unduly conservative assumption: the ratio between the entire tableau and the coin-tossing section (also known as the *mini tableau*) was held constant—specifically at a value of 2—rather than allowed to scale with l. This assumption will be relaxed in the present work, where we explore and leverage its dependence on l, the number of binary (nondeterministic) choices performed by the TM in question.

1.3. Results

We present and rigorously analyze two novel algorithms: `rFHB` and `sFHB`. Both are structured around the ψ_{trim}-based formulation given in formula (1), differing primarily in their final conjunct term, ψ^2_{extra}. Now, let $\widetilde{N}$ denote an $\langle N,k\rangle$ machine, as defined in this paper, with $k \in \{1,2\}$. The machine works with unary or binary notation, and is tasked with solving the `3-SAT` problem—using l binary choices on some input w of length n. We show that the runtime $R(l)$ of both `rFHB` and

 doi:10.20944/preprints202507.2367.v1

`sFHB`, when applied to $\widetilde{N}$ and w, admits a tight upper bound, effectively rendering the original `FHB` algorithm obsolete. Moreover, `sFHB` is significantly simpler to teach and implement than `rFHB`, offering conceptual clarity.

1.4. Outline

This article is structured into three primary sections: Orientation (Sections 2–4), Main Body (Sections 5–6), and Final Commentary with Analytical Addenda (Section 7 and Appendices A—F).

The Orientation spans 25 pages and presents a detailed yet approachable formalization of the `3-SAT` solver N^*. Key contributions include:

- Commentary on the construction of ψ^{η}_{step} using an extended tableau (Section 2)
- The introduction of *holes* within the extended tableau framework (Section 3)
- A complete definition of the solver N^* (Section 4)

The Main Body, comprising 18 pages, primarily investigates *long-range inter-cell dependencies*, culminating in the definition of ψ^{2}_{extra} (Section 5). It also introduces and analyzes two key algorithms: `rFHB` and `sFHB` (Section 6).

The article concludes with a Final Commentary and a set of Analytical Addenda. Section 7 offers reflective insights. Appendix A presents literature-based definitions and theorems; Appendix B introduces definitions specific to ψ^{1}_{extra}; Appendix C illustrates a long-range top-down constraint; Appendices D and F each offer a standard solution to a recurrence relation; and Appendix E provides an alternative proof of Theorem 3, the central result of the paper.

Remark 2. *Portions of the wording in the Orientation overlap with the author's earlier works [1,2]. The author retains full ownership — including commercial rights — of the prior content. As such, there are no legal constraints on reusing portions of those works in the current article.*

2. Extending the Tableau with Labels: Explicating ψ^{η}_{Step}

Consider an arbitrary nondeterministic TM N that, given an input w of length n, decides whether to accept or reject w in at most n^k steps for some constant k. We contemplate the behavior of a hypothetical accepting computation path p of N on an extended input $\hat{w}$, with

$$w \;=\; w_0 w_1 \ldots w_{n-1}, \qquad\qquad \hat{w} \;=\; w\square\square\ldots\square,$$

where the blank symbol $\square$ occurs $n^k - n$ times.

Path p depends on the execution of instructions, which can be uniquely labeled, such as:

$$t_{abc} : (q_1, a) \to \{(q_2, b, -), (q_3, c, +)\}.$$

This nondeterministic instruction, labeled t_{abc}, can be split into two deterministic ones:

$$t_{ab} : (q_1, a) \to (q_2, b, -), \qquad\qquad t_{ac} : (q_1, a) \to (q_3, c, +).$$

Each deterministic instruction is assigned a unique label (e.g., t_{ab}).

Instruction t_{ab} specifies that when N is in state q_1 and reading symbol a, the machine is supposed to transition to state q_2, rewrite the symbol a as b, and the tape head should move one cell to the left ($-$). A plus sign ($+$) indicates a unary move to the right.

In this section we focus on the formulation

$$\psi^{\eta}_{step} \;=\; {}^{*}\psi^{\eta}_{step} \;\wedge\; \psi^{\mathtt{det}}_{step}.$$

 doi:10.20944/preprints202507.2367.v1

We begin in Section 2.1 with a conventional account of nondeterminism. Section 2.2 then presents our own formulation, grounded in ${}^{*}\psi^{\eta}_{step}$. The core insights of this formulation are unpacked in Section 2.3. Finally, Section 2.4 explores the structure and role of $\psi^{\mathtt{det}}_{step}$.

2.1. Textbook Approach

To capture the step-by-step behavior of N on $\hat{w}$, we focus on the aforementioned instruction t_{abc} as it applies to the following TM configuration, denoted as C:

$$\ldots \quad a \quad \frac{a}{q_1} \quad d \quad \ldots$$

The symbol $\frac{a}{q_1}$ indicates that the machine is currently in state q_1, with its head oriented towards the tape cell containing the symbol a. This information can be expressed propositionally through the Boolean variable $x_{i,j,\frac{a}{q_1}}$, where indices i and j denote the row i and column j in a tableau—a matrix of n^k rows and n^k+2 columns, as shown in Table 1.

We analyze the execution of instructions t_{ab} and t_{ac} separately—both outcomes are depicted on the left and right sides of Table 2, respectively—before combining them into one implication. This results in an expression of the form:

$$C_1 \wedge C_2 \wedge C_3 \;\rightarrow\; T_{ab} \vee T_{ac}, \tag{4}$$

where both T_{ab} and T_{ac} take the form $C_1 \wedge C_2 \wedge C_3$. Ultimately, this forms a 3cnf-formula corresponding to the notion of a 2×3 window. By taking conjunctions over all 2×3 windows defined by N, and for each row and column in the tableau, we derive a 3cnf-formula ψ_{step} of size $O(n^{2k})$.

Table 1. A tableau: an $n^k \times \left(n^k+2\right)$ matrix. All cells in the leftmost column contain the boundary marker $\vdash$. Likewise for the rightmost column and the marker $\dashv$.

$\vdash$	$\frac{w_0}{q_0}$	w_1	w_2	$\ldots$	w_{n-1}	$\square$	$\ldots$	$\square$	$\dashv$
$\vdash$									$\dashv$
$\vdash$									$\dashv$
$\vdash$									$\dashv$

Table 2. Illustrating two 2×3 windows. The effect of instruction t_{ab} is shown on the left, while the effect of instruction t_{ac} is displayed on the right, with both illustrations read from top to bottom. Column indices range from $j-1$ to $j+1$. Each vertical arrow represents a change in precisely one symbol. Notably, $\frac{a}{q_1}$ is considered a single symbol, not two separate symbols.

a	$\frac{a}{q_1}$	d
$\downarrow$	$\downarrow$	
$\frac{a}{q_2}$	b	d
$j-1$	j	$j+1$

a	$\frac{a}{q_1}$	d
	$\downarrow$	$\downarrow$
a	c	$\frac{d}{q_3}$
$j-1$	j	$j+1$

To the best of our knowledge, every approach to $\mathcal{NP}$-completeness ultimately hinges on the notion of "a tableau," a concept that can be traced back to Cook's seminal paper [7]. The work of Cook in the United States was mirrored by Levin's concurrent developments in the Soviet Union [8,9].

 doi:10.20944/preprints202507.2367.v1

Specifically, the two tableau illustrations presented in Table 2 are modified adaptations of the exemplars found in Sipser [10](p. 280). (Sipser's textbook treatment uses "$q_1\ a$" instead of "$\frac{a}{q_1}$" when referring to a 2×3 window [10](p. 280). However, this is merely a cosmetic variation on the concept at hand.) Similarly, Papadimitriou introduces the notion of a "computation table" in Section 8.2 of his work [11]. In the same spirit, Hopcroft, Motwani, and Ullman refer to a comparable structure as "an array of cell/ID facts" [12](p. 443). Aaronson also echoes this idea of a tableau, albeit using more informal language in his accessible book [13](p. 61).

Table 3. The effect of instruction t_{ab} is shown on the left, while the effect of t_{ac} is displayed on the right. In both illustrations, the rows are arranged sequentially from the top row, indexed as $3l-2$, to the bottom row, indexed as $3l+1$. Each vertical arrow represents a change in precisely one symbol. If label t_{ab} is stored in $cell[3l, j]$ of the tableau—with row index $3l$ and column index j, where l and j are natural numbers—then we denote this with propositional variable $x_{3l,j,t_{ab}}$. Source: [2](p. 15).

$3l-2$	a	$\frac{a}{q_1}$	d
		$\downarrow$	
$3l-1$	a	t_{ab}	d
	$\downarrow$		
$3l$	$\frac{a}{q_2}$	t_{ab}	d
		$\downarrow$	
$3l+1$	$\frac{a}{q_2}$	b	d
	$j-1$	j	$j+1$

$3l-2$	a	$\frac{a}{q_1}$	d
		$\downarrow$	
$3l-1$	a	t_{ac}	d
			$\downarrow$
$3l$	a	t_{ac}	$\frac{d}{q_3}$
		$\downarrow$	
$3l+1$	a	c	$\frac{d}{q_3}$
	$j-1$	j	$j+1$

2.2. Our Approach

Can the step-by-step behavior of N on $\hat{w}$ be represented using a compact Horn formula, ${}^*\psi^{\eta}_{step}$, instead of a 3cnf-formula, ψ_{step}? We answer affirmatively in [2] by introducing an *extended tableau* with $3n^k+1$ rows and n^k+2 columns, explicitly storing the instruction labels, such as t_{ab} and t_{ac}. Two parts of such a tableau are shown in Table 3, illustrating only one change occurring at a time. This contrasts with the two simultaneous changes depicted in each illustration in Table 2.

We arrived at this result by adopting Aaronson's vision of philosophy as a "scout" that explores and maps out "intellectual terrain for science to *later* move in on, and build condominiums on ..." [13](p. 6, original emphasis). Building on this metaphor, and in dialogue with the perspectives of Dean [14], Tall [15], and Turner [16], our investigation explores the interplay between two distinct modes of reasoning: Aristotelian, step-by-step thinking and Platonic, static reasoning—as largely formulated by Linnebo and Shapiro [17].

These contrasting perspectives are illustrated in the following two quotes:

- Lance Fortnow as an Aristotelian:

  > A Turing machine has a formal definition but that's not how I think of it. When I write code, or prove a theorem involving computation, I feel the machine processing step by step. ...I feel it humming along, updating variables, looping, branching, searching until it arrives as its final destination and gives an answer. (Quoted from Lance Fortnow's blog post [18].)

- Robin K. Hill as a Platonist:

  > A Turing Machine is a static object, a declarative, a quintuple or septuple of the necessary components. The object δ that constitutes the transition function that describes the action is itself a set of tuples. All of this is written in appropriate symbols, and just sits there. (Quoted from Robin K. Hill's CACM blog post [19].)

 doi:10.20944/preprints202507.2367.v1

In our published work [2], we analyze these two intellectual modes in the context of nondeterministic TMs, and ultimately show how to transform the 3cnf-formula ψ_{step}, which captures the step-by-step behavior of N on $\hat{w}$, into the compact Horn formula ${}^*\psi^{\eta}_{step}$.

Technically, we employ an *extended tableau*—also called a *tableau with labels*. Here, the TM configurations are represented in rows $3l-2$ where $1 \leq l \leq n^k+1$. The two auxiliary rows, $3l-1$ and $3l$, each contain exactly one instruction label, and row $3l$ contains precisely one $\frac{s}{q}$ symbol, where s is a tape symbol and q a state symbol. Corresponding to Table 3, we define in [2] the Horn formula ${}^*\psi^{\eta}_{step}$ of size $O(n^{3k})$ literals.

Remark 3. *The formula ${}^*\psi^{\eta}_{step}$ is called ϕ^{η}_{step} in [2].*

The innovation behind ${}^*\psi^{\eta}_{step}$ is representing the binary choice between t_{ab} and t_{ac} as a conjunction of two formulas:

$$(\; x_{3l,j,t_{ab}} \; \rightarrow \; \bigwedge_i U_i \;) \quad \wedge \quad (\; x_{3l,j,t_{ac}} \; \rightarrow \; \bigwedge_i V_i \;),$$

yielding a Horn formula. In contrast, Sipser's textbook treatment expresses this choice with a disjunction, recall formula (4), which necessitates a 3cnf-formula.

To be more precise, $\bigwedge_i U_i$ represents the knowledge derived from $x_{3l,j,t_{ab}}$ through both upward and downward reasoning in our extended tableau. We express this derivation with the following formula:

$$x_{3l,j,t_{ab}} \; \rightarrow \bigwedge_{i \in \{3l-2,\, 3l-1,\, 3l+1\}} U_i,$$

which is equivalent to:

$$(x_{3l,j,t_{ab}} \; \rightarrow \; U_{3l-2}) \quad \wedge \quad (x_{3l,j,t_{ab}} \; \rightarrow \; U_{3l-1}) \quad \wedge \quad (x_{3l,j,t_{ab}} \; \rightarrow \; U_{3l+1}),$$

where U_i is a placeholder for a literal. The formula is a Horn formula. Likewise for t_{ac} and derived knowledge $\bigwedge_i V_i$, which amounts to:

$$x_{3l,j,t_{ac}} \; \rightarrow \; \bigwedge_i V_i,$$

where the subscript i stands for: $i \in \{3l-2,\, 3l-1,\, 3l+1\}$.

To convey the essence of our prior contribution without providing formal definitions, let $K(i,j,t)$ denote the knowledge derived from $x_{i,j,t}$ for some instruction label t of machine N stored in $cell[i,j]$, with $i = 3l$. In [2], we demonstrate the construction of multiple Horn formulas, including:

$$\begin{aligned}
\psi^1_V &= \ldots, \\
\psi^2_V &= \bigwedge_l \bigwedge_j \bigwedge_t \Big(x_{3l,j,t} \rightarrow K(3l,j,t) \Big), \\
\psi^3_V &= \ldots, \\
\psi_V &= \psi^1_V \wedge \psi^2_V \wedge \psi^3_V, \\
\psi_H &= \ldots, \\
{}^*\psi^{\eta}_{step} &= \psi_V \wedge \psi_H,
\end{aligned}$$

where the latter formula represents N's step-by-step behavior. We use "V" to denote "vertical" reasoning within the extended tableau, and "H" to signify "helicopter" reasoning across a block of rows, ranging from $3l-2$ to $3l+1$.

 doi:10.20944/preprints202507.2367.v1

2.3. Explicating ${}^*\psi^{\eta}_{Step}$

More rigorously, consider an arbitrary nondeterministic polynomial time TM $\langle N, k\rangle$, or N for short. Let the tape alphabet Φ, state set Q, and label set $T[\,]$ be extracted from the specifications of machine N.

Definition 1. *A* **nondeterministic polynomial time Turing machine**, *denoted as* $\langle N, k\rangle$, *is defined as* $N = (Q, \Gamma, \Phi, \delta, T, q_0, q_{accept}, q_{reject})$, *a nondeterministic Turing machine in accordance with Definition A5, which serves as a decider with a running time of* n^k *— as specified in Definition A8, where n and k represent the length of input w and some constant, respectively.*

Remark 4. *Without loss of generality, the nondeterminism associated with TM N consists solely of binary choices. For each such choice, say between instructions* t_1 *and* t_2, *the movement of* t_1 *is to the left* $(-)$, *while the movement of* t_2 *is to the right* $(+)$.

Recall that the propositional formula ψ_{trim} is defined as:

$$\psi_{trim} \;=\; \psi^{\eta}_{step} \wedge \psi_{start} \wedge \psi_{accept} \wedge \psi'_{cell} \wedge \psi^{1}_{extra} \wedge \psi^{2}_{extra},$$

with

$$\psi^{\eta}_{step} \;=\; {}^*\psi^{\eta}_{step} \;\wedge\; \psi^{\mathtt{det}}_{step}. \tag{5}$$

To elucidate the variables within ψ_{trim}, we define:

$$\Sigma \;=\; \Phi \cup (\Phi \times Q) \cup T[\,] \cup \{\vdash, \dashv\}.$$

For each i and j ranging from 1 to respectively $3n^k + 1$ and $n^k + 2$, and for every symbol s in Σ, we introduce a Boolean variable, $x_{i,j,s}$. We have a total of $O\left(n^{2k}\right)$ such variables.

The formula

$${}^*\psi^{\eta}_{step} \;=\; \psi_V \wedge \psi_H$$

reflects the coordination between the V and H subsystems. This coordination is achieved primarily by ensuring that specific vertical symbol conversions in the extended tableau are carried out in two distinct stages.

For instance, rather than directly converting symbol $\frac{a}{q_0}$ into symbol a' when traversing a column in the extended tableau top-down, the V subsystem first transforms $\frac{a}{q_0}$ into the intermediate label t_0, and only then into the symbol a'. This deliberate two-step conversion guarantees that V produces a unique intermediate trace—namely, the instruction label t_0 of machine N—which can then be identified by the H subsystem. This example, involving the label t_0, corresponds to the following deterministic machine instruction:

$$t_0 : (q_0, a) \to (q_1, a', +).$$

In general however, an instruction of N is nondeterministic. For each binary choice of N, such as

$$t_{abc} : (q_1, a) \to \{(q_2, b, -), (q_3, c, +)\},$$

we must first *determinize* the instruction by splitting it into two distinct deterministic ones:

$$t_{ab} : (q_1, a) \to (q_2, b, -), \qquad t_{ac} : (q_1, a) \to (q_3, c, +).$$

Each deterministic instruction is assigned a unique label (e.g., t_{ab}). Notably, determinizing an instruction that is already deterministic—such as t_0, t_{ab}, or t_{ac}—has no effect.

After applying *determinization* to all uniquely labeled instructions of N, we ensure that V, when selecting any deterministic instruction label t, explicitly records the label t as an intermediate trace

 doi:10.20944/preprints202507.2367.v1

in the extended tableau. Examples of t_{ab} and t_{ac} are shown in the center column, in the left and right illustrations, respectively, in Table 3. Consequently, H reads label t from the tableau and acts accordingly. The behavior of V and H is described by Horn formulas ψ_V and ψ_H, respectively, as formally defined in [2, Section 4].

Fundamentally, any conversion between two distinct tape symbols, say from a to b, in any column of the extended tableau, must occur through an intermediate trace. Table 4 provides an illustration, relying on the label t'_{ab} and, more precisely, the following instruction of machine N:

$$t'_{ab} : (q_3, a) \to (q_4, b, -).$$

The marked symbol a in the top row in Table 4 can only change into the marked symbol b in the bottom row via an intermediate trace, such as t'_{ab}.

A few additional clarifications regarding Table 4 are necessary. First, each symbol change from row to row is indicated with an arrow for better visualization. Second, the boxes surrounding symbols a and b are merely included to improve readability.

Table 4. A conversion from symbol a (marked in the top row) to b (marked in the bottom row). Row indices range from $3l-2$ to $3l+4$. Column indices from 2 to 5. Source: [2](p. 6, original emphasis]).

$3l-2$	a	$\frac{a}{q_1}$	$\boxed{a}$	b	
		$\downarrow$			
$3l-1$	a	t_{ac}	a	b	
			$\downarrow$		
$3l$	a	t_{ac}	$\frac{a}{q_3}$	b	
		$\downarrow$			
$3l+1$	a	c	$\frac{a}{q_3}$	b	$3l+1$
			$\downarrow$		
	a	c	t'_{ab}	b	$3l+2$
		$\downarrow$			
	a	$\frac{c}{q_4}$	t'_{ab}	b	$3l+3$
			$\downarrow$		
	a	$\frac{c}{q_4}$	$\boxed{b}$	b	$3l+4$
	2	3	4	5	

To summarize, the novelty of our approach in [2] is twofold. First, we introduce an *extended tableau* that explicitly stores instruction labels, enabling single-symbol changes between consecutive rows. Second, we analyze the tableau from both a vertical perspective (ψ_V) and a helicopter perspective (ψ_H), combining them into the succinct Horn formula: ${}^*\psi^{\eta}_{step} = \psi_V \wedge \psi_H$.

2.4. Explicating $\psi^{\mathtt{det}}_{Step}$

We now explicate the second conjunct of equation (5), which pertains exclusively to the deterministic instructions t of the nondeterministic TM under consideration; specifically, those satisfying $t \in T_{\mathtt{det}}$, as defined in Definition A7.

We distinguish between the subsets $T^{+}_{\mathtt{det}}$ and $T^{-}_{\mathtt{det}}$, along with their associated formulas $\psi^{\mathtt{det}}_{+}$ and $\psi^{\mathtt{det}}_{-}$, respectively. The overall formula for deterministic transitions is thus expressed as:

$$\psi^{\mathtt{det}}_{step} \quad = \quad \psi^{\mathtt{det}}_{+} \ \wedge \ \psi^{\mathtt{det}}_{-}. \tag{6}$$

 doi:10.20944/preprints202507.2367.v1

We present the explicit definition of the first conjunct, $\psi_+^{\mathtt{det}}$, leaving the construction of $\psi_-^{\mathtt{det}}$ to the reader by symmetry:

$$\psi_+^{\mathtt{det}} = \bigwedge_l \bigwedge_j \bigwedge_{t \in T_{\mathtt{det}}^+} \bigwedge_{s'} \Big[\mathrm{source}(t)@(3l-2,j) \ \wedge \ s'@(3l-2,j+1) \ \rightarrow \ t@(3l,j) \ \wedge \ \mathrm{write}(t)@(3l+1,j) \ \wedge \ \frac{s'}{\mathrm{target}(t)}@(3l+1,j+1)\Big],$$

where $1 \leq l \leq n^k+1$ and $1 \leq j \leq n^k+2$, with $s' \in \Phi$. For precise definitions of the operators such as $\mathrm{source}(t)$, $\mathrm{write}(t)$, and $\mathrm{target}(t)$, see Definition A6.

The formula $\psi_+^{\mathtt{det}}$ comprises $O(n^{2k})$ literals. An analogous construction, along with the same complexity bound, applies to $\psi_-^{\mathtt{det}}$. Together, these formulas encapsulate traditional top-down reasoning. Here, the top is row $3l-2$, going via row $3l$ to the bottom row $3l+1$. This aligns with the established result that such reasoning can be fully captured by a Horn formula [20](p. 35).

3. Extended Tableau with Holes

An extended tableau is a matrix consisting of $(3n^k+1) \times (n^k+2)$ cells. It is formed by augmenting each of the n^k rows of the basic tableau (except the bottom row) with two auxiliary rows. Going forward, the context will clarify which version of the tableau—extended or basic—is being referred to. The reader is expected to switch between these representations as appropriate.

Notation

Given an extended tableau, the cell at row i and column j is denoted by $cell[i,j]$ and is intended to store a symbol $s \in \Sigma$. The contents of these cells are represented using the variables of ψ_{trim}, which, unless specified, is used in place of the original formula ψ.

When the variable $x_{i,j,s}$ is assigned the value 1, it signifies that $cell[i,j]$ in the extended tableau contains the symbol s. We also denote this situation as follows:

$$s@(i,j).$$

Conversely, we write

$$\neg\, s@(i,j) \quad \text{or} \quad \overline{s@(i,j)}$$

when $x_{i,j,s}$ is assigned 0, using either notation interchangeably to improve readability.

When referring to the corresponding basic tableau, which consists of $n^k \times (n^k+2)$ cells, we use the notation

$$s@(i,j)_{\mathcal{B}}$$

to abbreviate

$$s@(3i-2,j),$$

indicating that the cell at row i and column j in the basic tableau corresponds to row $3i-2$ and column j in the extended tableau. Similarly, we write

$$\neg\, s@(i,j)_{\mathcal{B}} \quad \text{or} \quad \overline{s@(i,j)_{\mathcal{B}}}$$

to mean

$$\overline{s@(3i-2,j)}.$$

Definition 2. *Consider ψ_{trim} and, correspondingly, its extended and basic tableaux. We say that $cell[i,j]$ in the extended tableau contains a* hole *iff $x_{i,j,s}$ is false for all $s \in \Sigma$. We say that $cell[i,j]$ in the basic tableau contains a hole iff $cell[3i-2,j]$ in the extended tableau contains a hole.*

 doi:10.20944/preprints202507.2367.v1

Definition 2 does not exclude the possibility that a specific variable x_{i,j,s^*}, for some $s^* \in \Sigma$, may later be "turned on," effectively filling the hole with symbol s^*. For example, suppose $cell[i,j]$ initially contains a hole, but a human agent (e.g., a user of an off-the-shelf HORNSAT solver $\mathcal{H}$) later assigns it the tape-state symbol $\frac{a}{q_5}$. This scenario is illustrated in:

- Table 5 (with an extended tableau), and
- Table 6 (with the corresponding basic tableau).

What are the implications of the proposition $\frac{a}{q_5}@(3l-2,j)$, with $1 \leq l \leq n^k+1$?

To address this question in the remainder of this section, we first introduce the Horn formula ψ^1_{extra} in Section 3.1, followed by a closer examination of one of its subformulas, ψ^{extend}_{extra}, in Section 3.2. Then we focus on the concept of "user interaction" in Section 3.3. Finally, we present the original FHB (Filling Holes with Backtracking) algorithm in Section 3.4.

Table 5. Here, the proposition $\frac{a}{q_5}@(i,j)$ holds, with $i = 3l-2$. In words: $cell[i,j]$ in the extended tableau contains the tape-state symbol $\frac{a}{q_5}$.

	×	×	×	×	×						
	×	×	×	×	×	×					
	×	×	×	×	×	×	×				×
i	×	×	×	×	×	×	×	×	$\frac{a}{q_5}$	×	×
	×	×	×	×	×	×	×				×
	×	×	×	×	×	×					
	×	×	×	×	×						
	×	×	×	×							
	×	×	×								
	×	×									
	×										
									j		

 doi:10.20944/preprints202507.2367.v1

Table 6. Here, the proposition $\frac{a}{q_5}@(i^*, j)_{\mathcal{B}}$ holds. This means that the symbol $\frac{a}{q_5}$ appears at position (i^*, j) in the basic tableau (shown below), with $i^* = \frac{i+2}{3} = \frac{3l-2+2}{3} = l$.

	×	×	×	×	×						
	×	×	×	×	×	×					
	×	×	×	×	×	×	×				×
i^*	×	×	×	×	×	×	×	×	$\frac{a}{q_5}$	×	×
	×	×	×	×	×	×	×				×
	×	×	×	×	×	×					
	×	×	×	×	×						
	×	×	×	×							
	×	×	×								
	×	×									
	×										
									j		

Table 7. Filling cells (in the extended tableau) with additional crosses as a logical consequence of the state of affairs depicted in Table 5 and ψ^1_{extra}.

	×	×	×	×	×						
	×	×	×	×	×						
	×	×	×	×	×						
	×	×	×	×	×	×					
	×	×	×	×	×	×					
	×	×	×	×	×	×					
	×	×	×	×	×	×	×				×
	×	×	×	×	×	×	×				×
	×	×	×	×	×	×	×				×
i	×	×	×	×	×	×	×	×	$\frac{a}{q_5}$	×	×
	×	×	×	×	×	×	×				×
	×	×	×	×	×	×	×				×
	×	×	×	×	×	×	×				×
	×	×	×	×	×	×					
	×	×	×	×	×	×					
	×	×	×	×	×	×					
	×	×	×	×	×						
	×	×	×	×	×						
	×	×	×	×	×						
	×	×	×	×							
	×	×	×	×							
	×	×	×	×							
	×	×	×								
	×	×	×								
	×	×	×								
	×	×									
	×	×									
	×	×									
	×										
	×										
	×								j		

3.1. Introducing ψ^1_{extra}

The formula ψ^1_{extra} captures global properties of a TM computation. While it is redundant in the context of ψ's satisfiability, it proves useful when ψ_{trim} is satisfiable in the general case—namely when the basic tableau still contains holes. To convey the meaning of ψ^1_{extra}, we begin by examining the example in Table 5.

 doi:10.20944/preprints202507.2367.v1

Remark 5. *The Horn formula ψ^1_{extra} corresponds to ψ^{η}_{extra} from [1], with two notational differences:*

- *the formula ψ^{η}_{extra} is defined over a basic tableau, rather than an extended tableau; and*
- *ψ^{η}_{extra} uses the notation $q\ s$ (instead of $\frac{s}{q}$) to denote the TM's head is scanning s in state q.*

A precise translation from ψ^{η}_{extra} to ψ^1_{extra} is straightforward. We treat ψ^1_{extra} informally here and provide a formal definition in Appendix B. Importantly, ψ^1_{extra} is a Horn formula of size $O(n^{\kappa})$, where the constant $\kappa = 4k$.

If $\frac{a}{q_5}@(i,j)$ holds (where $a \in \Phi$, $q_5 \in Q$, and $i = 3l-2$), meaning $x_{i,j,\frac{a}{q_5}} = 1$, then the tape head of N in $cell[i,j]$ cannot reach any of the crossed-out cells in Table 5. This restriction follows from the fact that a TM can move its tape head by at most one cell per transition—either left or right. In terms of Table 5, this corresponds to a transition between rows $3l-2$ and $3l+1$, with $1 \leq l \leq n^k$—or equivalently, between any two consecutive rows marked with crosses. These transitions mirror those between consecutive rows in the basic tableau (Table 6).

Furthermore, in each column of the basic tableau (and similarly in the extended tableau), all crossed-out cells either contain or are required to contain the same tape symbol $s \in \Phi$. This constraint follows from the fact that a TM can only modify a tape symbol when its head is directly over that cell.

Therefore, filling the hole in $cell[3l-2,j]$ (Table 5) with the symbol $\frac{a}{q_5}$ effectively amounts to filling in all crossed-out cells, albeit indirectly. In other words, the condition $\frac{a}{q_5}@(3l-2,j)$ in Table 5 ensures that only the uncrossed cells in Table 6 can encode the binary choices made by N on w.

Formula ψ^1_{extra} expresses these restrictions as a conjunction of four parts:

$$\psi^1_{extra} \quad = \quad \bigwedge_{1\leq l\leq n^k+1} \ \bigwedge_{1\leq j\leq n^k+2} \left[\bigwedge_{s\in\Phi} \ \bigwedge_{q\in Q} \left[\psi^{single}_{extra} \ \wedge \ \bigwedge_{s'\in\Phi} \left[\ \psi^{left}_{extra} \ \wedge \ \psi^{right}_{extra} \ \right] \right] \ \wedge \ \psi^{extend}_{extra} \right]$$

The formal definition of each conjunct, provided in Appendix B, aligns with Table 7, which extends the structure shown in Table 5. In each column of Table 7, every cross represents the same tape symbol.

Each conjunct plays a distinct role:

- Single part (ψ^{single}_{extra}): Ensures that each row $3l-2$ contains at most one tape-state symbol $\frac{s}{q}$, such as $\frac{a}{q_5}$.
- Left part (ψ^{left}_{extra}): Handles the crossed-out cells to the left of $\frac{a}{q_5}@(3l-2,j)$ in Table 7.
- Right part (ψ^{right}_{extra}): Covers the crossed-out cells to the right of $\frac{a}{q_5}@(3l-2,j)$ in Table 7.
- Extend part (ψ^{extend}_{extra}): Introduces additional refinements not discussed in [1], as that work does not consider the ψ^{η}_{step}-based intricacies of an extended tableau.

The final point is elaborated in the following section.

3.2. Formula ψ^{extend}_{extra}

To illustrate the use of ψ^{extend}_{extra}, consider Table 3 again and suppose that proposition

$$\frac{a}{q_1}@(3l-2,j)$$

has already been guessed (by the human agent) for some fixed l and j. We present two clarifications.

3.2.1. Clarification 1

If it later follows that

$$\frac{a}{q_2}@(3l+1,j-1)$$

also holds—either due to another user guess or, more realistically, as a consequence of $cell[3l+1,j+1]$ containing a tape symbol—then

$$t_{ab}@(3l,j)$$

 doi:10.20944/preprints202507.2367.v1

should automatically hold as well. The current state of affairs corresponds to the left illustration in Table 3.

This inference arises from implications embedded in ψ_{extra}^{extend}, such as:

$$\frac{a}{q_1}@(3l-2,j) \ \wedge \ \frac{a}{q_2}@(3l+1,j-1) \quad \rightarrow \quad t_{ab}@(3l,j).$$

Similarly, if t_{ac} applies instead of t_{ab}, as shown with the right illustration in Table 3, we have:

$$\frac{a}{q_1}@(3l-2,j) \ \wedge \ \frac{d}{q_3}@(3l+1,j+1) \quad \rightarrow \quad t_{ac}@(3l,j).$$

3.2.2. Clarification 2

Resuppose that

$$\frac{a}{q_1}@(3l-2,j)$$

has been guessed for some fixed l and j and that, in adherence to the left illustration in Table 3, $cell[3l+1, j+1]$ contains some tape symbol s. Then the following family of inferences,

$$\bigwedge_{s\in\Phi}\bigwedge_{s'\in\Phi}\left[\frac{a}{q_1}@(3l-2,j) \ \wedge \ s@(3l+1,j+1) \ \wedge \ s'@(3l-2,j-1) \ \rightarrow \ \frac{s'}{q_2}@(3l+1,j-1)\right],$$

allows for an automatic derivation of

$$\frac{a}{q_2}@(3l+1,j-1),$$

where s and s' stand for d and a, respectively, and q_2 denotes the target state of t_{ab} (in our running example).

Remark 6. *The formal definition of ψ_{extra}^{extend} is provided in Appendix B and incorporates both clarifications discussed above. However, Clarification 2 also hinges upon the constraints in [2, Section 4.3].*

3.3. User Interaction

We assume that ψ_{trim} is satisfiable and that the extended tableau reflects this condition, typically containing several holes, as illustrated in Table 7. A hole in the (extended) tableau, located at row index i and column index j, represents more than just an empty cell. To be conservative, we stipulate the following:

- Single Hole: If $cell[i, j]$ is the only hole in the tableau, it corresponds to at most c possible accepting computation paths, where c is the cardinality of Σ. (In fact, in this scenario, it contributes to representing at most one accepting path.)
- Two Holes: If $cell[i, j]$ is one of two holes in the tableau, it contributes to representing up to $c \times c$ possible accepting computation paths.
- Three Holes: If $cell[i, j]$ is one of three holes, it contributes to representing up to $c \times c \times c$ possible accepting computation paths.

This pattern continues, with each additional hole multiplying the maximum number of possible accepting computation paths by c.

In the general case, the (extended) tableau, composed of a polynomial number of cells, indirectly represents an exponentially large number of paths for N on w, including paths that are syntactically inadmissible from the perspective of N's step-by-step behavior. Among the syntactically admissible paths, there are both rejecting and accepting paths.

 doi:10.20944/preprints202507.2367.v1

This flexibility is achieved by leaving most cells unfilled. The Horn clauses associated with ψ_{trim} remain implicitly active in the background, waiting for an external user to fill in a hole via an additional specification, such as

$$\frac{s^*}{q^*}@(i^*, j^*),$$

where

$$s^* \in \Phi \text{ and } q^* \in Q,$$
$$i^* = 3l^* - 2 \text{ and } 1 < l^* < n^k + 1,$$
$$1 < j^* < n^k + 2.$$

Consequently, the HORNSAT solver $\mathcal{H}$ is called upon again, now tasked with satisfying

$$\psi_{trim} \wedge \frac{s^*}{q^*}@(i^*, j^*),$$

which stands for

$$\psi_{trim} \wedge x_{i^*, j^*, \frac{s^*}{q^*}}.$$

After two more user interventions, the solver is tasked with satisfying the following type of formula:

$$\psi_{trim} \wedge \frac{s^*}{q^*}@(i^*, j^*) \wedge \frac{s^{**}}{q^{**}}@(i^{**}, j^{**}) \wedge \frac{s^{***}}{q^{***}}@(i^{***}, j^{***}),$$

where

$$i^{**} = 3l^{**} - 2,$$
$$i^{***} = 3l^{***} - 2.$$

However, if the user's guess (e.g., $\frac{s^*}{q^*}@(i^*, j^*)$) leads the solver $\mathcal{H}$ to detect unsatisfiability, backtracking is required. In such cases, the user may revise the assignment—for instance, replacing

$$\frac{s^*}{q^*}@(i^*, j^*) \quad \text{with} \quad \frac{s_*}{q_*}@(i^*, j^*),$$

where $(s^*, q^*) \neq (s_*, q_*)$. This means that $s^* \neq s_*$ or $q^* \neq q_*$ or both.

Fortunately, as shown in Claim 3 of [1] in the context of the basic tableau, asymptotic analysis (with $n \to \infty$) reveals that filling any hole in the center row of a convex polygon of holes reduces the space for binary choices by a factor of $\frac{1}{2}$. This effect is illustrated in two places:

- Table 8 demonstrates the initial scaling.
- Table 9 shows a second intervention in row 4, where enforcing $\frac{b}{q_9}@(4, 12)_{\mathcal{B}}$ results in 41 additional crossed-out cells.

 doi:10.20944/preprints202507.2367.v1

Table 8. Basic tableau: crossing out 113 cells out of $16 \times 16 = 256$ cells.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	×															
	×	×														×
	×	×	×												×	×
	×	×	×	×										×	×	×
5	×	×	×	×	×								×	×	×	×
	×	×	×	×	×	×						×	×	×	×	×
	×	×	×	×	×	×	×				×	×	×	×	×	×
8	×	×	×	×	×	×	×	×	$\frac{a}{q_5}$	×	×	×	×	×	×	×
	×	×	×	×	×	×	×				×	×	×	×	×	×
10	×	×	×	×	×	×						×	×	×	×	×
	×	×	×	×	×								×	×	×	×
	×	×	×	×										×	×	×
	×	×	×												×	×
	×	×														×
15	×															
16																

Table 9. Basic tableau: a second intervention in row 4 amounts to crossing out 41 boldfaced cells out of 63 cells in rows 1–7. The depicted scenario is ultimately unsatisfiable, as the downward movement from $\frac{b}{q_9}$ and the upward movement from $\frac{a}{q_5}$ will fail to converge harmoniously in the same cell.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	×	×	×	×	×	×	×	×								×
	×	×	×	×	×	×	×	×	×						×	×
	×	×	×	×	×	×	×	×	×	×				×	×	×
	×	×	×	×	×	×	×	×	×	×	×	$\frac{b}{q_9}$	×	×	×	×
5	×	×	×	×	×	×	×	×	×	×			×	×	×	×
	×	×	×	×	×	×	×	×	×			×	×	×	×	×
	×	×	×	×	×	×	×	×			×	×	×	×	×	×
8	×	×	×	×	×	×	×	×	$\frac{a}{q_5}$	×	×	×	×	×	×	×
	×	×	×	×	×	×	×				×	×	×	×	×	×
10	×	×	×	×	×	×						×	×	×	×	×
	×	×	×	×	×								×	×	×	×
	×	×	×	×										×	×	×
	×	×	×												×	×
	×	×														×
15	×															
16																

Remark 7. *To simplify our exposition, the leftmost columns in our depicted tableaux do not contain the boundary marker* $\vdash$*. However, strictly speaking, column* 1 *should contain the boundary marker* $\vdash$*, while tape and tape-state symbols appear only from column* 2 *onward.*

3.4. The FHB *Algorithm*

Conceptually, the FHB algorithm relies on a standard HORNSAT solver $\mathcal{H}$ and integrates the actions of the external user, automating her interventions and incorporating backtracking as a built-in feature. We remark that $\mathcal{H}$ operates in nearly linear time [21].

Recall the definition of ψ_{trim},

$$\psi_{trim} \quad = \quad \psi^{\eta}_{step} \wedge \psi_{start} \wedge \psi_{accept} \wedge \psi'_{cell} \wedge \psi^{1}_{extra} \wedge \psi^{2}_{extra},$$

 doi:10.20944/preprints202507.2367.v1

which will turn out to be $O(n^{\kappa_0})$ literals long for some constant κ_0. For the time being, we substitute the truth value 1 for ψ^2_{extra} (to be revisited in Section 5). In this context, ψ^1_{extra} is the largest conjunct, of size $O(n^{4k})$, as follows from Appendix B.

Additionally, we will have at most $O(n^k)$ extra stipulations of the form $\frac{s}{q}@(3l-2, j)$, namely, one per three rows in the extended tableau. Hence, an upper bound on the total cost of our HORNSAT instance,

$$\psi_{trim} \quad \wedge \quad \frac{s^*}{q^*}@(3l^*-2, j^*) \quad \wedge \quad \ldots,$$

can be expressed as

$$p(n) = \kappa_1 \cdot n^{\kappa_2},$$

for some constants κ_1 and κ_2.

The FHB algorithm begins with ψ_{trim}, an instance of size $< p(n)$, and thus $\leq p(n)$, and runs the solver $\mathcal{H}$ on it, resulting in a trivial "satisfiable" as a tentative outcome. (If N's computation on w is deterministic, then the outcome is permanent and either satisfiable or unsatisfiable.) Next, the algorithm selects the center row, or one of the two center rows, of the basic tableau and injects the first tape-state symbol—the first $\frac{s}{q}$ symbol appearing in a standard list representation of $\Phi \times Q$—into the leftmost hole in that row. If backtracking is required, subsequent iterations will use different tape-state symbols, and if this does not suffice, the next hole (from left to right) in the row will be filled instead, starting again with the first $\frac{s}{q}$ symbol appearing in a standard ordering of $\Phi \times Q$, and so on.

For a row containing holes, there are at most n^k ways to inject some specific tape-state symbol $\frac{s}{q}$, with $(s, q) \in \Phi \times Q$, into that row. This leads to our key observation:

> There are at most $c_0 \cdot n^k$ ways to inject any tape-state symbol $\frac{s}{q}$, with $(s, q) \in \Phi \times Q$, into a row, where c_0 denotes the cardinality of $\Phi \times Q$.

The first user intervention results in scaling the space of binary choices by $\frac{1}{2}$, shrinking from size $\leq p(n)$ to size $\leq \Delta \cdot p(n)$, with $\Delta = \frac{1}{2}$. In the next two interventions, our algorithm selects the middle row (or, if applicable, one of the two middle rows) of the first and second convex polygons of holes, read from top to bottom in the basic tableau. In the next four interventions, our algorithm selects the middle row (or one of two middle rows) of each of the four smaller convex polygons of holes, moving sequentially from top to bottom. This pattern continues in subsequent steps.

Immediately after each intervention, the FHB algorithm directs the solver $\mathcal{H}$ to check the entire (extended) tableau for unsatisfiability and, in the process, simplify the underlying Horn clauses as much as possible, taking into account all constraints specified by $\psi_{trim} \wedge \ldots$, where the dots refer to the cumulative intervention stipulations made up to that point.

After each stage of interventions—one intervention in stage 1, two interventions in stage 2, four interventions in stage 3, eight interventions in stage 4, and so on—the solver $\mathcal{H}$ runs on an instance that has been shrunk in size by $\Delta = \frac{1}{2}$. To be technically precise, the solver $\mathcal{H}$ continues operating on the entire instance, but the space of binary choices has been shrunk by a factor of Δ after each stage. As a result, we are *intrinsically* dealing with an instance of size $\leq \Delta^m \cdot p(n)$ after m stages. Hence, m is bounded from above by $O(log(n))$. Additionally, the bookkeeping for backtracking itself incurs at most a polynomial cost. A runtime stack with a constant overhead per recursive call is sufficient in practice [22].

Remark 8. *In future work, the software engineer could reduce the exponent κ_2 in*

$$p(n) = \kappa_1 \cdot n^{\kappa_2},$$

by considering the on-the-fly generation of the Horn constraints associated with ψ^{η}_{step} and/or ψ^1_{extra}.

For instance, the formula ψ^{η}_{step} could expand and contract based on the placement of the $\frac{s}{q}$ symbols in the tableau, rather than conservatively accounting for all possibilities in advance. Similarly, the tailored constraints

 doi:10.20944/preprints202507.2367.v1

related to ψ^1_{extra} could be added only when a guess $\frac{s^*}{q^*}$@$(3l^* - 2, j^*)$ is made, causing the formula ψ^1_{extra} to grow incrementally with each additional guess and shrinking during backtracking.

Theorem 1. *(Reappropriated from Daylight [1](p. 27).) Consider a nondeterministic polynomial time TM $\langle N, k\rangle$ that runs on an input w of length n. The runtime $R(n)$ of the* FHB *algorithm, applied to $\langle N, k\rangle$ and w, satisfies*

$$R(n) \quad \leq \quad 2^{\kappa \cdot n^k},$$

where the constant $\kappa > 1$.

Theorem 1 suggests a prohibitive runtime. However, a refinement of the method leads to a significantly tighter upper bound, as discussed in the remainder of this paper.

4. The 3-SAT Solver N^*

Even a devil's advocate would have to admit that the analysis thus far is unduly pessimistic, as it assumes that *every* cell in the basic tableau could involve a binary nondeterministic guess. In reality, the situation is considerably more favorable. Only a portion of the basic tableau entails binary choices, and crucially, the outcome of each guess (i.e., the transition to either state q_a or q_b) determines the presence of a specific state symbol (e.g., q_c) in another cell—typically further down—in the basic tableau. These observations about the basic tableau naturally carry over to the extended tableau.

To clarify why the situation is more favorable, we begin in this section by formally defining the 3-SAT solver N^*, supplemented by informal insights. In Section 5, we explore the inter-cell dependencies of N^* across distant rows of the basic tableau. Then, in Section 6, we conclude with a presentation of two algorithms: the refined rFHB and the streamlined sFHB algorithm.

Overview

The TM N^* runs on an input word w of length n, where a substring $\widetilde{w}$ of input word w encodes a 3-SAT formula ϕ with l propositional variables $(x_1, \ldots, x_l)$ and m clauses. Each clause consists of three literals—for example, $x_2 \vee \neg x_7 \vee x_{92}$. Here, $l \leq m < n$.

With respect to 3-SAT itself, an informal grasp of the following stipulations suffices:

- Each variable x_i appears in at least two literals across the formula ϕ; otherwise, such a variable (ocurring only once) can be eliminated through preprocessing.
- No clause contains the same variable x more than once—whether as $x \vee x$, $x \vee \overline{x}$, or $\overline{x} \vee \overline{x}$. Hence, $l \leq m$.
- Accordingly, we assume that as m increases sufficiently, so does l.
  - The lower bound is $m = \Omega(l)$, as each variable must be constrained in some way.
  - In a deliberately conservative upper bound, we posit $m = O(l^3)$.

TM N^* is an $\langle N, 2\rangle$ machine. Its input word w has the form:

$$\widetilde{w}\#\square\square \ldots \square\#,$$

where the number of blank symbols ($\square$) in between the two # markers is exactly l, and the comma is included solely for readability.

The operation of N^*, with state set Q and tape alphabet Φ, proceeds through three sequential stages:

1. Coin-Tossing Stage $\widehat{S}$ (Section 4.1)
2. Updating Stage $\overleftrightarrow{S}$ (Section 4.2)
3. Checking Stage $\overleftarrow{S}$ (Section 4.3)

- Coin-Tossing Stage $\widehat{S}$: To denote elements specific to this stage, we annotate instruction labels and state symbols with a "roof" symbol $\widehat{\ }$. This annotation signifies *divergence*—that is, the

 doi:10.20944/preprints202507.2367.v1

nondeterminism which arises exclusively during this stage. The coin-tossing stage uses seven machine instructions labeled $\widehat{t_1}, \widehat{t_2}, \ldots, \widehat{t_7}$ and operates with the following state set:

$$\widehat{Q} = \{\widehat{q_0}, \widehat{q_1}, \widehat{Q_0}, \} \quad \text{with} \quad \widehat{Q} \subset Q.$$

- Updating Stage $\overleftrightarrow{S}$: Here, instruction labels and state symbols are annotated with a bidirectional arrow $\leftrightarrow$, indicating the machine's back-and-forth traversal across the tape. This movement is generally required to update the encoding of the formula ϕ (i.e., the substring $\widetilde{w}$) based on the coin-toss outcomes. The updating stage employs instructions labeled $\overleftrightarrow{t_1}$, $\overleftrightarrow{t_2}$, ..., and uses the following set of states:

$$\overleftrightarrow{Q} = \{\overleftrightarrow{q_0}, \overleftrightarrow{q_1}, \overleftrightarrow{q_2}, \overleftrightarrow{Q_0}, \overleftrightarrow{U_0}, \ldots\} \quad \text{with} \quad \overleftrightarrow{Q} \subset Q.$$

- Checking Stage $\overleftarrow{S}$: In this final stage, instruction labels and state symbols are annotated with a left arrow $\leftarrow$, reflecting the machine's predominant leftward movement. While most transitions are leftward, occasional rightward steps will occur locally. The checking stage uses instructions labeled $\overleftarrow{t_1}$, $\overleftarrow{t_2}$, ... and the state set:

$$\overleftarrow{Q} = \{\overleftarrow{q_3}, \overleftarrow{q_0}, \overleftarrow{Q_0}, \overleftarrow{q_{reject}}, \ldots\} \quad \text{with} \quad \overleftarrow{Q} \subset Q.$$

The symbol $\overleftarrow{q_{reject}}$ serves as a surrogate for q_{reject}, and we adopt the latter notation in the remainder of this paper.

4.1. Coin-Tossing Stage

The machine N^* stores the encoding of the formula ϕ, represented as the string $\widetilde{w}$, on its tape. The tape head is initially positioned at the first blank symbol, $\square$, right after $\widetilde{w}\#$. More specifically, the initial tape configuration is as follows:

$$\widetilde{w}\# \underset{\widehat{q_1}}{\square} \square \ldots \square \#. \tag{7}$$

Here, N^* is in state $\widehat{q_1}$, reading the first of l blank symbols. The punctuation is included solely for readability.

The machine generates l bits, proceeding from left to right and writing each bit—either 0 or 1—into a separate tape cell. This sequence, which is supposed to represent the outcome of l independent coin tosses, is enclosed at both ends by the marker #.

One outcome of any coin toss must correspond to a rightward movement ($+$) and the other to a leftward movement ($-$). To enforce this constraint—consistent with Remark 4—we implement the following behavior, starting in the $\widehat{q_1}$ tossing state:

$$\widehat{t_1} : (\widehat{q_1}, \square) \to (\widehat{q_1}, 1, +) \qquad \widehat{t_2} : (\widehat{q_1}, \square) \to (\widehat{q_0}, 0, -)$$

Among all instructions pertaining to N^*, only $\widehat{t_1}$ and $\widehat{t_2}$ involve nondeterministic choices.

- If a coin toss yields bit 1, the machine moves its head one cell to the right and re-enters the $\widehat{q_1}$ state. See instruction $\widehat{t_1}$.
- If a coin toss yields bit 0, the machine first moves its head one cell to the left and enters state $\widehat{q_0}$. See instruction $\widehat{t_2}$. Then the machine performs two deterministic moves to the right, ending up in state $\widehat{q_1}$ again.
  - See instructions $\widehat{t_3} - \widehat{t_5}$ for the first move to the right:

$$\widehat{t_3} : (\widehat{q_0}, \#) \to \left(\widehat{Q_0}, \#, +\right) \qquad \widehat{t_4} : (\widehat{q_0}, 0) \to \left(\widehat{Q_0}, 0, +\right) \qquad \widehat{t_5} : (\widehat{q_0}, 1) \to \left(\widehat{Q_0}, 1, +\right)$$

 doi:10.20944/preprints202507.2367.v1

– See instruction $\widehat{t_6}$ for the second move to the right:

$$\widehat{t_6} : \left(\widehat{Q_0}, 0\right) \rightarrow (\widehat{q_1}, 0, +)$$

- Once the machine reaches the rightmost # marker (while in state $\widehat{q_1}$), it moves leftward and enters state $\overleftrightarrow{q_2}$:

$$\widehat{t_7} : (\widehat{q_1}, \#) \rightarrow (\overleftrightarrow{q_2}, \#, -)$$

Upon completing the coin-tossing process, the machine will have generated l bits,

$$b_1, b_2, \ldots, b_{l-1}, b_l,$$

for, respectively, the propositional variables:

$$x_l, x_{l-1}, \ldots, x_2, x_1.$$

In other words, the j-th coin toss from the right ($1 \leq j \leq l$) determines the truth assignment, 0 or 1, for propositional variable x_j.

Assigning the truth value 1 to the variable x_j entails that, during the Updating Stage $\overleftrightarrow{S}$, the machine will set each encoded occurrence of x_j in the word $\widetilde{w}$ to 1, and each encoded occurrence of $\neg x_j$ to 0. (A similar remark holds for the truth value 0.)

By preprogramming the proper constraints, to be detailed in Section 5, filling holes with tape-state symbols in the coin-tossing section of the basic tableau will automatically propagate to filling corresponding holes with tape-state symbols in lower sections of the entire basic tableau. Moreover, if and when all l coins have been tossed, the remaining basic tableau—and therefore the entire basic tableau—is fully determined. (Once the basic tableau is fully determined, the extended tableau is as well.) Even a devil's advocate would expect this property to be reflected in a worst-case analysis of the FHB algorithm or a refinement thereof.

4.1.1. Four Coin Tosses

Table 10 illustrates the structure of the coin-tossing process for $l = 4$. The hyphens and dots in the illustration represent potential positions of the tape head (i.e., tape-state symbols), with the distinction between the two serving only for visual clarity. The two extreme computation runs are depicted solely with hyphens: the diagonal run at the top (consisting of five hyphens) produces all four bits as 1, while the zigzagging run takes longer to complete and results in all four bits being 0.

 doi:10.20944/preprints202507.2367.v1

Table 10. The coin-tossing section of the basic tableau, also known as the basic *mini tableau* [1]: tossing four coins from left to right, resulting in the truth values for variables x_4, x_3, x_2, and x_1. The row $\widehat{r}$ corresponds to the state attained after all $l = 4$ coins have been tossed and when the rightmost column $\widehat{c} = 6$ is reached. In this case, l is even. Consequently, $\widehat{r}$ is odd and belongs to the set $\widehat{r} \in \{5, 7, 9, 11, 13\} = \{l + 1, l + 3, \ldots, 3l + 1\}$.

1	#	—				#
2	—		—			
3		—		—		
4			—		—	
5		—		·		—
6			—		·	
7				—		·
8			—		·	
9				—		·
10					—	
11				—		·
12					—	
13						—
	1	2	3	4	5	6

As Table 10 conveys, the coin-tossing process is represented by a matrix with $3l + 1$ rows and $l + 2$ columns. This matrix can be embedded within a basic $(3l + 1) \times (3l + 3)$ *mini tableau*. The corresponding extended $(3 \cdot (3l + 1) + 1) \times (3l + 3)$ *mini tableau* is not shown here.

Theorem 1 in Section 3.4 provides a basis for analyzing the runtime associated with the *mini tableau*. Crucially, if the 3-SAT solver N^* were *solely* responsible for tossing l coins, then no tighter bound than 2^l—akin to Theorem 1's worst-case runtime of the FHB algorithm—applies. In reality, however, the coin tosses of N^* are made to correlate through the word $\widetilde{w}$, which encodes the 3-SAT formula ϕ. For not all sequences of l coin tosses are valid—if any at all.

4.1.2. Properties of Computation

To appreciate (and ultimately formalize) the correlation between the l coin tosses, we begin by presenting two basic insights regarding the computation runs of N^* on w:

1. The basic *mini tableau*—and, more rigorously, the extended *mini tableau*—captures all nondeterminism (coin tossing) inherent to N^*, while also including rote deterministic computations.
   - Example 1: If four 1 bits are tossed consecutively, N^*'s tape head lands on $cell[5, 6]$ of Table 10 and immediately begins rote deterministic computation from row 6.
   - Example 2: If four 0 bits are tossed instead, N^* uses the entire *mini tableau* to complete the coin tossing, reaching $cell[13, 6]$, before starting rote deterministic computation in row 14 onward.
2. The rote deterministic computation does not revisit column 6 (in Table 10) or any column to its right. More formally, the rightmost column $\widehat{c}$ of the basic *mini tableau*, which is of length $3l + 1$, contains exactly one tape-state symbol (namely, $\frac{\#}{\widehat{q}_1}$). Furthermore, in the rest of the basic tableau, the same column $\widehat{c}$ contains only the # tape symbol and thus no other tape-state symbols.
   (a) Although the two hyphens and three dots in column $\widehat{c}$ indicate multiple possible positions for a tape-state symbol, only one tape-state symbol can appear in any particular computation. Moreover, that symbol must be $\frac{\#}{\widehat{q}_1}$, implying that the following proposition holds:
   $$\frac{\#}{\widehat{q}_1} @ (\widehat{r}, \widehat{c})_{\mathcal{B}},$$
   for a suitable row index $\widehat{r}$.
   (b) For each input formula ϕ and any valid placement of $\frac{\#}{\widehat{q}_1}$ in the rightmost column $\widehat{c}$, which amounts to specifying $\widehat{r}$, we can determine (and thus preprogram) the position of the

 doi:10.20944/preprints202507.2367.v1

machine's head—though not the corresponding tape-state symbol—in *every* subsequent row of the entire basic tableau. In other words, the implication of guessing row index $\widehat{r}$—with $\widehat{r} \in \{5, 7, 9, 11, 13\}$ in Table 10—is as follows:

> Apart from a substantial portion of the basic *mini tableau*, each row from $\widehat{r}+1$ onward in the basic tableau contains exactly one uncrossed cell—indicating that the machine's tape head is restricted from occupying the crossed-out cells.

This property is ensured through the construction of N^*, detailed further in Sections 4.2–4.3.

4.2. Updating Stage

Upon entering Stage $\overleftrightarrow{S}$, the machine N^* has its tape head positioned at the rightmost bit, immediately before the rightmost # marker. The tape configuration is as follows:

$$\& \, \$ \, cl_m \, \$ \ldots \$ \, cl_2 \, \$ \, cl_1 \, \$ \, \# \, 1 \, 0 \ldots 0 \, \overset{0}{\underset{\overleftrightarrow{q_2}}{}} \, \#,$$

with

$$\overset{0}{\underset{\overleftrightarrow{q_2}}{}} @ (\widehat{r}+1, \widehat{c}-1)_{\mathcal{B}}.$$

The machine is in state $\overleftrightarrow{q_2}$, with the head reading the bit 0 (in the current example).

Observe that the leftmost & symbol marks the beginning of the string $\widetilde{w}$, and that the rightmost \$ symbol marks the end of $\widetilde{w}$. Moreover, each encoded clause, such as cl_1, is delimited by the \$ marker.

4.2.1. Unary Encoding

To simplify the control flow, we adopt a unary encoding for each variable and its negation. Specifically, a variable x_α is encoded as a string of α copies of the symbol a:

$$a \, a \, \ldots a.$$

Similarly, the negated variable $\neg x_\alpha$ is encoded as a string of α copies of a distinct symbol $\overline{a}$:

$$\overline{a} \, \overline{a} \, \ldots \overline{a}.$$

Remark 9. *Encoding l propositional variables in binary requires $l \times (\log l)$ bits, whereas a unary encoding requires $l \times l$ bits. Although this results in an increase, it does not amount to an exponential blow-up; furthermore, it remains acceptable within the scope of computability theory [23].*

Rather than stating formal desiderata, the unary encoding scheme is best illustrated through a few examples. Suppose Clause 1 is as follows:

$$(x_3 \vee \neg x_2 \vee x_1).$$

Its unary encoding, denoted as cl_1, is:

$$a \, a \, a \, \vee \, \overline{a} \, \overline{a} \, \vee \, a \vee,$$

where the comma and extra spacing are included solely to enhance readability.

The subscript of the variable indicates the number of occurrences of either the symbol a or $\overline{a}$. If the literal is positive, the symbol a is used, if the literal is negative, $\overline{a}$ is used.

Now consider Clause 2:

$$(\neg x_4 \vee x_2 \vee \neg x_3).$$

 doi:10.20944/preprints202507.2367.v1

Its unary encoding, denoted cl_2, is:

$$\overline{a}\,\overline{a}\,\overline{a}\,\overline{a} \vee a\,a \vee \overline{a}\,\overline{a}\,\overline{a} \vee .$$

The punctuation is added solely for clarity in this exposition.

4.2.2. Updating a Unary Encoding: Part 1

Let us revisit Clause 1 and its unary encoding, cl_1:

$$a\,a\,a \vee \overline{a}\,\overline{a} \vee a \vee .$$

Suppose the variable x_1 is assigned the truth value 1:

$$x_1 = 1.$$

Then, after an update operation, the coresponding string encoding should take the following form:

$$???\vee?? \vee 1\vee,$$

where each question mark (?) denotes a "don't care" entry—a meta-symbol whose assigned value (a specific symbol from the alphabet Σ) is inconsequential within this context.

Similarly, if

$$x_2 = 1,$$

then, eventually, the coresponding encoding should take the form:

$$??? \vee 0?\vee? \vee .$$

Finally, if

$$x_3 = 0,$$

the encoding should ultimately take the form:

$$0??\vee??\vee?\vee,$$

again with question marks indicating "don't care" entries.

In summary, for each encoded literal, the leftmost symbol $\sigma \in \{a, \overline{a}\}$ should be overwritten with the appropriate truth value. The remaining symbols a and $\overline{a}$ in the encoded literal are superfluous.

4.2.3. Updating a Unary Encoding: Part 2

Suppose now that the machine's tape head approaches the encoded clause cl_1:

$$a\,a\,a \vee \overline{a}\,\overline{a} \vee a\vee,$$

from the right, under the assumption that $x_1 = 1$. The machine then traverses the entire (encoded) clause from right to left, continuing leftward across all (encoded) clauses preceding cl_1. During this traversal, the machine updates the first symbol $\sigma \in \{a, \overline{a}\}$ of each encoded literal—in every (encoded) clause—according to the following rule:

- If $\sigma = a$, overwite it with 1.
- If $\sigma = \overline{a}$, overwite it with 0.

Applying this rule to cl_1 yields the updated encoding:

$$a\,a\,1 \vee \overline{a}\,0 \vee 1 \vee .$$

 doi:10.20944/preprints202507.2367.v1

We now proceed to the second update pass, under the assumption that $x_2 = 1$. The machine's tape head approaches the previously updated string from the right-hand side, traversing the entire clause from right to left, as well as all (encoded) clauses to its left. In this pass, the machine updates the second symbol $\sigma \in \{a, \overline{a}\}$ of each encoded literal—in every (encoded) clause—according to the following rule:

- If $\sigma = a$, overwite it with 1.
- If $\sigma = \overline{a}$, overwite it with 0.

Applying this rule, we obtain the updated string:

$$a\,1\,1 \vee 0\,0 \vee 1 \vee .$$

Finally, we arrive at the third update pass, under the assumption that $x_3 = 0$. The machine's tape head once again approaches the previously updated string from the right-hand side, traversing the entire clause from right to left, along with all (encoded) clauses to its left. In this pass, it updates the third symbol $\sigma \in \{a, \overline{a}\}$ of each encoded literal—in every (encoded) clause—according to the following rule:

- If $\sigma = a$, overwite it with 0.
- If $\sigma = \overline{a}$, overwite it with 1.

Applying this rule yields the final updated string:

$$0\,1\,1 \vee 0\,0 \vee 1 \vee .$$

The resulting string has the desired form

$$0\,?\,? \vee 0\,? \vee 1 \vee,$$

corresponding to the assignment

$$x_3 = 0 \text{ and } x_2 = x_1 = 1,$$

which is best read from right to left, in keeping with the direction of the tape head's movement.

4.2.4. The Updating Code

We shall now present the corresponding instructions of N^*. Upon entering the Updating Stage $\overleftrightarrow{S}$, the machine N^* has the configuration

$$\&\,\$\,cl_m\,\$ \ldots \$\,cl_2\,\$\,a\,a\,a \vee \overline{a}\,\overline{a} \vee a \vee \$\,\#\,1\,0 \ldots 0\,\underset{\overleftrightarrow{q_2}}{0}\,\#,$$

and starts with $\overleftrightarrow{t_1}$ or $\overleftrightarrow{t_2}$:

$$\overleftrightarrow{t_1} : (\overleftrightarrow{q_2}, 1) \to (\overleftrightarrow{q_1}, \#, -), \qquad \overleftrightarrow{t_2} : (\overleftrightarrow{q_2}, 0) \to (\overleftrightarrow{q_0}, \#, -).$$

In both cases, the bit (1 or 0) is overwritten with the # symbol.

In our running example, instruction $\overleftrightarrow{t_1}$ does not apply, whereas $\overleftrightarrow{t_2}$ does—since the scanned bit is not 1, but 0. This yields:

$$\&\,\$\,cl_m\,\$ \ldots \$\,cl_2\,\$\,a\,a\,a \vee \overline{a}\,\overline{a} \vee a \vee \$\,\#\,1\,0 \ldots \underset{\overleftrightarrow{q_0}}{0}\,\#\,\#.$$

Due to the symmetry in the behavior of updating a 1-bit and a 0-bit, we restrict our analysis to the second case ($\overleftrightarrow{t_2}$). Accordingly, the instructions and labels presented below are not exhaustive; additional instructions exist but are omitted here for brevity.

 doi:10.20944/preprints202507.2367.v1

Moving Left

Instructions $\overleftrightarrow{t_3}$ and $\overleftrightarrow{t_4}$ allow the machine to remain in state $\overleftrightarrow{q_0}$ while traversing all remaining bits from right to left:

$$\overleftrightarrow{t_3} : (\overleftrightarrow{q_0}, 0) \to (\overleftrightarrow{q_0}, 0, -) \qquad \overleftrightarrow{t_4} : (\overleftrightarrow{q_0}, 1) \to (\overleftrightarrow{q_0}, 1, -)$$

These transitions preserve the bit values and simply move the tape head to the left.

Eventually, upon encountering the leftmost # marker, the machine transitions from state $\overleftrightarrow{q_0}$ to $\overleftrightarrow{Q_0}$:

$$\overleftrightarrow{t_5} : (\overleftrightarrow{q_0}, \#) \to \left(\overleftrightarrow{Q_0}, \#, -\right),$$

resulting in the following configuration:

$$\& \$\, cl_m \, \$ \ldots \$\, cl_2 \, \$\, a\, a\, a \vee \overline{a}\, \overline{a} \vee a \vee \overset{\$}{\overleftrightarrow{Q_0}} \#1 0 \ldots 0 \# \#.$$

When the machine encounters the \$ symbol, or when it later visits the symbols a or $\overline{a}$, it simply continues moving left, remaining in state $\overleftrightarrow{Q_0}$:

$$\overleftrightarrow{t_{6,x}} : \left(\overleftrightarrow{Q_0}, x\right) \to \left(\overleftrightarrow{Q_0}, x, -\right), \quad x \in \{\$, a, \overline{a}\}.$$

The result of executing the instruction $\overleftrightarrow{t_{6,\$}}$ for the first time is as follows:

$$\& \$\, cl_m \, \$ \ldots \$\, cl_2 \, \$\, a\, a\, a \vee \overline{a}\, \overline{a} \vee a \overset{\vee}{\overleftrightarrow{Q_0}} \$ \# 1 0 \ldots 0 \# \#.$$

The machine has now reached the first (encoded) literal of an (encoded) clause and switches from state $\overleftrightarrow{Q_0}$ to state $\overleftrightarrow{U_0}$, moving left:

$$\overleftrightarrow{t_7} : \left(\overleftrightarrow{Q_0}, \vee\right) \to \left(\overleftrightarrow{U_0}, \vee, -\right),$$

resulting in the configuration:

$$\& \$\, cl_m \, \$ \ldots \$\, cl_2 \, \$\, a\, a\, a \vee \overline{a}\, \overline{a} \vee \overset{a}{\overleftrightarrow{U_0}} \vee \$ \# 1 0 \ldots 0 \# \#.$$

While in the update state $\overleftrightarrow{U_0}$, encountering any of the symbols $\vee$, \$, 0, or 1 does not trigger a state change; the machine's tape head simply continues moving left:

$$\overleftrightarrow{t_{8,x}} : \left(\overleftrightarrow{U_0}, x\right) \to \left(\overleftrightarrow{U_0}, x, -\right), \quad x \in \{\vee, \$, 0, 1\}.$$

However, upon reading the symbol a (or $\overline{a}$), the machine carries out the update by writing the bit 0 (or 1, respectively) and returning to state $\overleftrightarrow{Q_0}$:

$$\overleftrightarrow{t_9} : \left(\overleftrightarrow{U_0}, a\right) \to \left(\overleftrightarrow{Q_0}, 0, -\right), \qquad \overleftrightarrow{t_{10}} : \left(\overleftrightarrow{U_0}, \overline{a}\right) \to \left(\overleftrightarrow{Q_0}, 1, -\right).$$

In our running example, we obtain:

$$\& \$\, cl_m \, \$ \ldots \$\, cl_2 \, \$\, a\, a\, a \vee \overline{a}\, \overline{a} \overset{\vee}{\overleftrightarrow{Q_0}} 0 \vee \$ \# 1 0 \ldots 0 \# \#.$$

At this point, also reconsider the instructions $\overleftrightarrow{t_{6,x}}$, where $x \in \{\$, a, \overline{a}\}$: the machine remains in state $\overleftrightarrow{Q_0}$ while scanning from right to left in search of the next $\vee$ symbol, if one exists. In our running example, the next $\vee$ symbol has already been located.

 doi:10.20944/preprints202507.2367.v1

Remark 10. *As previously mentioned, we are restricting our analysis to the case of updating a 0-bit. However, for future reference, we provide the twin instructions of* $\overleftrightarrow{t_9}$ *and* $\overleftrightarrow{t_{10}}$*, as follows:*

$$\overleftrightarrow{\tau_9} : \left(\overleftrightarrow{U_1}, a\right) \to \left(\overleftrightarrow{Q_1}, 1, -\right), \qquad \overleftrightarrow{\tau_{10}} : \left(\overleftrightarrow{U_1}, \bar{a}\right) \to \left(\overleftrightarrow{Q_1}, 0, -\right),$$

which update a and $\bar{a}$ *to, respectively,* 1 *and* 0*.*

Moving Right

Finally, if the machine is in state $\overleftrightarrow{U_0}$ or $\overleftrightarrow{Q_0}$ and encounters the leftmost marker &, it transitions to state $\overleftrightarrow{h}$ and begins moving right:

$$\overleftrightarrow{t_{11}} : \left(\overleftrightarrow{U_0}, \&\right) \to \left(\overleftrightarrow{h}, \&, +\right),$$

$$\overleftrightarrow{t_{12}} : \left(\overleftrightarrow{Q_0}, \&\right) \to \left(\overleftrightarrow{h}, \&, +\right),$$

resulting in the following configuration:

$$\& \overset{\$}{\underset{\overleftrightarrow{h}}{}} cl_m \$ \ldots \$ cl_2 \$ a\, a\, 0 \vee \bar{a}\, 1 \vee 0 \vee \$ \# 1 0 \ldots 0 \# \#.$$

In state $\overleftrightarrow{h}$, the machine scans rightward, searching for the leftmost occurrence of the symbol #. It continues moving right over the symbols $\vee$, \$, 0, 1, a, and $\bar{a}$ without changing state:

$$\overleftrightarrow{t_{13,x}} : \left(\overleftrightarrow{h}, x\right) \to \left(\overleftrightarrow{h}, x, +\right), \quad x \in \{\vee, \$, 0, 1, a, \bar{a}\}.$$

This ultimately brings us to, e.g., the following configuration:

$$\& \$ cl_m \$ \ldots \$ cl_2 \$ a\, a\, 0 \vee \bar{a}\, 1 \vee 0 \vee \$ \overset{\#}{\underset{\overleftrightarrow{h}}{}} 1 0 \ldots 0 \# \#.$$

Upon encountering #, the machine transitions from state $\overleftrightarrow{h}$ to $\overleftrightarrow{i}$ and continues moving right:

$$\overleftrightarrow{t_{14}} : \left(\overleftrightarrow{h}, \#\right) \to \left(\overleftrightarrow{i}, \#, +\right),$$

which results in one of the following two types of configurations:

$$\& \$ cl_m \$ \ldots \$ cl_2 \$ 0 1 0 \vee 0 1 \vee 0 \vee \$ \# \overset{\#}{\underset{\overleftrightarrow{i}}{}} \# \ldots \# \# \#,$$

and

$$\& \$ cl_m \$ \ldots \$ cl_2 \$ a\, a\, 0 \vee \bar{a}\, 1 \vee 0 \vee \$ \# \overset{1}{\underset{\overleftrightarrow{i}}{}} 0 \ldots 0 \# \#.$$

In the first type of configuration, when there are no remaining toss outcomes (to the right of the leftmost # symbol), the machine transitions from state $\overleftrightarrow{i}$ to $\overleftarrow{q_3}$, and moves left:

$$\overleftrightarrow{t_{15}} : \left(\overleftrightarrow{i}, \#\right) \to \left(\overleftarrow{q_3}, \#, -\right).$$

This transition marks the beginning of the final phase: the Checking Stage.

However, in the second type of configuration, when one or more toss outcomes are still present, the machine continues moving right, searching for the rightmost bit. It first transitions from $\overleftrightarrow{i}$ to $\overleftrightarrow{j}$ while preserving the bit value:

$$\overleftrightarrow{t_{16}} : \left(\overleftrightarrow{i}, 0\right) \to \left(\overleftrightarrow{j}, 0, +\right), \qquad \overleftrightarrow{t_{17}} : \left(\overleftrightarrow{i}, 1\right) \to \left(\overleftrightarrow{j}, 1, +\right).$$

 doi:10.20944/preprints202507.2367.v1

Once in state $\overleftrightarrow{j}$, the machine continues moving right through any remaining bits:

$$\overleftrightarrow{t_{18}} : \left(\overleftrightarrow{j}, 0\right) \to \left(\overleftrightarrow{j}, 0, +\right), \qquad \overleftrightarrow{t_{19}} : \left(\overleftrightarrow{j}, 1\right) \to \left(\overleftrightarrow{j}, 1, +\right).$$

When the leftmost # of the remaining # symbols is encountered, the machine transitions from state $\overleftrightarrow{j}$ to state $\overleftrightarrow{q_2}$ and moves left:

$$\overleftrightarrow{t_{20}} : \left(\overleftrightarrow{j}, \#\right) \to \left(\overleftrightarrow{q_2}, \#, -\right).$$

At this point, the machine has returned to state $\overleftrightarrow{q_2}$, scanning the rightmost bit:

$$\& \$ \, cl_m \, \$ \ldots \$ \, cl_2 \, \$ \, a \, a \, 0 \vee \bar{a} \, 1 \vee 0 \vee \$ \# 1 0 \ldots \overset{0}{\overleftrightarrow{q_2}} \# \#.$$

It is now prepared to repeat the earlier steps, thereby continuing the Updating Stage for another iteration.

4.3. Checking Stage

Once all tossed coins have been processed, N^* reaches—via instruction $\overleftrightarrow{t_{15}}$—the following configuration:

$$\& \$ \, cl_m \, \$ \ldots \$ \, cl_2 \, \$ 0 1 0 \vee 0 1 \vee 0 \vee \$ \overset{\#}{\overleftarrow{q_3}} \# \ldots \# \#,$$

Here, N^* is ready to perform a final traversal of the tape—from its current position to the left, with local left-to-right movements—to check whether each instantiated clause is satisfied.

Remark 11. *The caveat is that we want the tape movement behavior to remain invariant—that is, independent of the contents of the (encoded) instantiated clauses.*

The initial phase of the right-to-left traversal is governed by the following two instructions:

$$\overleftarrow{t_1} : (\overleftarrow{q_3}, \#) \to (\overleftarrow{q_0}, \#, -), \qquad \overleftarrow{t_2} : (\overleftarrow{q_0}, \$) \to (\overleftarrow{q_0}, \$, -),$$

which results in the following configuration:

$$\& \$ \, cl_m \, \$ \ldots \$ \, cl_2 \, \$ 0 1 0 \vee 0 1 \vee 0 \overset{\vee}{\overleftarrow{q_0}} \$ \# \# \ldots \# \#.$$

While in state $\overleftarrow{q_0}$, the machine switches to state $\overleftarrow{Q_0}$ upon encountering the first $\vee$ symbol in the (encoded and instantiated) clause currently being scanned:

$$\overleftarrow{t_3} : (\overleftarrow{q_0}, \vee) \to \left(\overleftarrow{Q_0}, \vee, -\right),$$

resulting, in our example, in the following configuration:

$$\& \$ \, cl_m \, \$ \ldots \$ \, cl_2 \, \$ 0 1 0 \vee 0 1 \vee \overset{0}{\overleftarrow{Q_0}} \vee \$ \# \# \ldots \# \#.$$

The states $\overleftarrow{Q_0}$, $\overleftarrow{Q_{00}}$, and $\overleftarrow{Q_{000}}$ indicate that the machine is currently processing—right to left—the first, second, and third (encoded and instantiated) literals of the clause being scanned, respectively. As we shall see shortly, transitions from these states to $\overleftarrow{R_0}$, $\overleftarrow{R_{00}}$, and $\overleftarrow{R_{000}}$, respectively, signify that the machine has located the leftmost bit of the corresponding literal, thereby determining its truth value.

 doi:10.20944/preprints202507.2367.v1

4.3.1. First Literal

In state $\overleftarrow{Q_0}$, the machine continues moving left through the (encoded and instantiated) clause, remaining in the same state as it reads bits:

$$\overleftarrow{t_{4,x}} : \left(\overleftarrow{Q_0}, x\right) \to \left(\overleftarrow{Q_0}, x, -\right), \quad x \in \{0,1\}.$$

Upon reaching the next $\vee$ symbol, the machine switches to the result state $\overleftarrow{R_0}$ and moves one cell to the right:

$$\overleftarrow{t_5} : \left(\overleftarrow{Q_0}, \vee\right) \to \left(\overleftarrow{R_0}, \vee, +\right),$$

resulting in the following configuration:

$$\& \$\, cl_m \,\$ \ldots \$\, cl_2 \,\$ 0\,1\,0 \vee 0\,1 \vee \underset{\overleftarrow{R_0}}{0} \vee \$\#\# \ldots \#\#.$$

The machine can now determine whether the clause is already satisfied or not:

$$\overleftarrow{t_6} : \left(\overleftarrow{R_0}, 1\right) \to (\overleftarrow{s_{00}}, 1, -), \qquad \overleftarrow{t_7} : \left(\overleftarrow{R_0}, 0\right) \to (\overleftarrow{q_{00}}, 0, -),$$

where the letter "s" in $\overleftarrow{s_{00}}$ stands for "satisfied."

In our running example, the first literal is not satisfied, resulting in the following configuration:

$$\& \$\, cl_m \,\$ \ldots \$\, cl_2 \,\$ 0\,1\,0 \vee 0\,1 \underset{\overleftarrow{q_{00}}}{\vee} 0 \vee \$\#\# \ldots \#\#.$$

Since the first literal is not satisfied, the machine proceeds in state $\overleftarrow{q_{00}}$ (rather than $\overleftarrow{s_{00}}$). As we shall see shortly, the machine subsequently transitions from $\overleftarrow{q_{00}}$ to $\overleftarrow{Q_{00}}$.

4.3.2. Second Literal: Part 1

The states $\overleftarrow{q_{00}}$ and $\overleftarrow{Q_{00}}$ indicate that the machine is currently processing—right to left—the second (encoded and instantiated) literal of the clause being scanned. The transition from $\overleftarrow{Q_{00}}$ to $\overleftarrow{R_{00}}$ signifies that the machine has located the leftmost bit of the literal, thereby determining its truth value.

The relevant transitions are as follows:

$$\overleftarrow{t_8} : (\overleftarrow{q_{00}}, \vee) \to \left(\overleftarrow{Q_{00}}, \vee, -\right).$$

$$\overleftarrow{t_{9,x}} : \left(\overleftarrow{Q_{00}}, x\right) \to \left(\overleftarrow{Q_{00}}, x, -\right), \quad x \in \{0,1\}.$$

$$\overleftarrow{t_{10}} : \left(\overleftarrow{Q_{00}}, \vee\right) \to \left(\overleftarrow{R_{00}}, \vee, +\right).$$

Once in the result state $\overleftarrow{R_{00}}$,

$$\& \$\, cl_m \,\$ \ldots \$\, cl_2 \,\$ 0\,1\,0 \vee \underset{\overleftarrow{R_{00}}}{0}\, 1 \vee 0 \vee \$\#\# \ldots \#\#,$$

the machine inspects the bit:

- If the truth value is 1, it switches to state $\overleftarrow{s_{000}}$.
- If the truth value is 0, it switches to state $\overleftarrow{q_{000}}$.

These transitions are captured by:

$$\overleftarrow{t_{11}} : \left(\overleftarrow{R_{00}}, 1\right) \to (\overleftarrow{s_{000}}, 1, -), \qquad \overleftarrow{t_{12}} : \left(\overleftarrow{R_{00}}, 0\right) \to (\overleftarrow{q_{000}}, 0, -),$$

where the letter "s" in $\overleftarrow{s_{000}}$ stands for "satisfied."

 doi:10.20944/preprints202507.2367.v1

In our running example, the second literal is not satisfied, resulting in:

$$\&\,\$\,cl_m\,\$\ldots\$\,cl_2\,\$\,0\,1\,0\,\underset{\overleftarrow{q_{000}}}{\vee}\,0\,1\;\vee\;0\;\vee\;\$\#\#\ldots\#\#.$$

Since the second literal is not satisfied, the machine proceeds in state $\overleftarrow{q_{000}}$ (rather than $\overleftarrow{s_{000}}$). As we shall see shortly, the machine subsequently transitions from $\overleftarrow{q_{000}}$ to $\overleftarrow{Q_{000}}$.

4.3.3. Second Literal: Part 2

Similar to $\overleftarrow{q_{00}}$ and $\overleftarrow{Q_{00}}$, the states $\overleftarrow{s_{00}}$ and $\overleftarrow{S_{00}}$ also indicate that the machine is currently processing—right to left—the second (encoded and instantiated) literal of a clause. However, in this case, the clause is already known to be satisfied:

$$\&\,\$\,cl_m\,\$\ldots\$\,cl_2\,\$\,0\,1\,0\;\vee\;0\,1\,\underset{\overleftarrow{s_{00}}}{\vee}\,1\;\vee\;\$\#\#\ldots\#\#.$$

In conformity with Remark 11, the transition from $\overleftarrow{S_{00}}$ to $\overleftarrow{T_{00}}$ indicates that the machine has located the leftmost bit of the literal.

The corresponding transitions are:

$$\overleftarrow{t'_8} : (\overleftarrow{s_{00}}, \vee) \to \left(\overleftarrow{S_{00}}, \vee, -\right).$$

$$\overleftarrow{t'_{9,x}} : \left(\overleftarrow{S_{00}}, x\right) \to \left(\overleftarrow{S_{00}}, x, -\right), \quad x \in \{0,1\}.$$

$$\overleftarrow{t'_{10}} : \left(\overleftarrow{S_{00}}, \vee\right) \to \left(\overleftarrow{T_{00}}, \vee, +\right).$$

Regardless of the leftmost bit's value, the machine proceeds to state $\overleftarrow{s_{000}}$:

$$\overleftarrow{t'_{11}} : \left(\overleftarrow{T_{00}}, 1\right) \to (\overleftarrow{s_{000}}, 1, -), \qquad \overleftarrow{t'_{12}} : \left(\overleftarrow{T_{00}}, 0\right) \to (\overleftarrow{s_{000}}, 0, -).$$

4.3.4. Third Literal: Part 1

The states $\overleftarrow{q_{000}}$ and $\overleftarrow{Q_{000}}$ indicate that the machine is currently processing—right to left—the third (encoded and instantiated) literal of the clause being scanned. The transition from $\overleftarrow{Q_{000}}$ to $\overleftarrow{R_{000}}$ signifies that the machine has located the leftmost bit of the literal, thereby determining its truth value.

The relevant transitions are as follows:

$$\overleftarrow{t_{13}} : (\overleftarrow{q_{000}}, \vee) \to \left(\overleftarrow{Q_{000}}, \vee, -\right).$$

$$\overleftarrow{t_{14,x}} : \left(\overleftarrow{Q_{000}}, x\right) \to \left(\overleftarrow{Q_{000}}, x, -\right), \quad x \in \{0,1\}.$$

$$\overleftarrow{t_{15}} : \left(\overleftarrow{Q_{000}}, \$\right) \to \left(\overleftarrow{R_{000}}, \$, +\right).$$

Once in the result state $\overleftarrow{R_{000}}$,

$$\&\,\$\,cl_m\,\$\ldots\$\,cl_2\,\$\,\underset{\overleftarrow{R_{000}}}{0}\,1\,0\;\vee\;0\,1\;\vee\;0\;\vee\;\$\#\#\ldots\#\#,$$

the machine inspects the bit:

- If the truth value is 1, it switches to state $\overleftarrow{q_0}$.
- If the truth value is 0, it switches to state q_{reject}.

 doi:10.20944/preprints202507.2367.v1

These transitions are captured by:

$$\overleftarrow{t_{16}} : \left(\overleftarrow{R_{000}}, 1\right) \to (\overleftarrow{q_0}, 1, -), \qquad \overleftarrow{t_{17}} : \left(\overleftarrow{R_{000}}, 0\right) \to (q_{reject}, 0, -).$$

In our running example, the third literal is not satisfied, and we obtain:

$$\& \$\, cl_m \,\$ \ldots \$\, cl_2 \underset{q_{reject}}{\$} 0\,1\,0 \vee 0\,1 \vee 0 \vee \$\#\# \ldots \#\#.$$

4.3.5. Third Literal: Part 2

Similar to $\overleftarrow{q_{000}}$ and $\overleftarrow{Q_{000}}$, the states $\overleftarrow{s_{000}}$ and $\overleftarrow{S_{000}}$ also indicate that the machine is currently processing—right to left—the third (encoded and instantiated) literal of a clause. However, in this case, the clause is already known to be satisfied. In conformity with Remark 11, the transition from $\overleftarrow{S_{000}}$ to $\overleftarrow{T_{000}}$ indicates that the machine has located the leftmost bit of the literal.

The corresponding transitions are:

$$\overleftarrow{t'_{13}} : (\overleftarrow{s_{000}}, \vee) \to \left(\overleftarrow{S_{000}}, \vee, -\right).$$

$$\overleftarrow{t'_{14,x}} : \left(\overleftarrow{S_{000}}, x\right) \to \left(\overleftarrow{S_{000}}, x, -\right), \quad x \in \{0,1\}.$$

$$\overleftarrow{t'_{15}} : \left(\overleftarrow{S_{000}}, \$\right) \to \left(\overleftarrow{T_{000}}, \$, +\right).$$

Regardless of the leftmost bit's value, the machine proceeds to state $\overleftarrow{q_0}$:

$$\overleftarrow{t'_{16}} : \left(\overleftarrow{T_{000}}, 1\right) \to (\overleftarrow{q_0}, 1, -), \qquad \overleftarrow{t'_{17}} : \left(\overleftarrow{T_{000}}, 0\right) \to (\overleftarrow{q_0}, 0, -).$$

4.3.6. Early Rejection

The symbol q_{reject} must never appear in any cell of the (extended) tableau—this follows directly from the semantics of the Horn formula ψ_{accept} (see point 2 in the introduction of this paper). Arguably for the sake of mathematical aesthetics, one could handle its hypothetical presence similarly to $\overleftarrow{t_2}$, using the transition:

$$\overleftarrow{t_{18}} : (q_{reject}, \$) \to (\overleftarrow{q_0}, \$, -).$$

However, we prefer to *discard* instruction $\overleftarrow{t_{18}}$ from further consideration.

4.3.7. Looping on the Left

If, and only if, all (encoded and instantiated) clauses have been examined from right to left, the tape head enters a loop confined to the two leftmost (non-blank) cells of the tape. In combination with instruction $\overleftarrow{t_2}$, the following instruction

$$\overleftarrow{t_{19}} : (\overleftarrow{q_0}, \&) \to (\overleftarrow{q_0}, \&, +)$$

ensures that the tape head oscillates indefinitely between these two leftmost cells associated with the word $\widetilde{w}$.

5. Long-Range Inter-Cell Dependencies: Defining ψ^2_{extra}

Consider a devil's advocate who adheres to the formal definition of the 3-SAT solver N^* presented in Section 4. She will recognize that, for a given ϕ, once the row $\widehat{r}$ corresponding to the guess

$$\underset{\widehat{q_1}}{\#}@(\widehat{r}, \widehat{c})_{\mathcal{B}},$$

 doi:10.20944/preprints202507.2367.v1

is fixed, the positions of tape-state symbols in all subsequent rows of the basic tableau become fully determined. Furthermore, only the first $\widehat{r}$ rows of the basic tableau involve binary (nondeterministic) choices.

Based on this insight and the notion of "long-range inter-cell dependency," introduced in this section, we aim to improve the prohibitive runtime established by Theorem 1, as revisited in Section 6. To this end, we first introduce the concept of a *coin-tossing scenario* (Section 5.1), followed by an examination of two distinct forms of long-range inter-cell dependencies within the basic tableau of N^*: top-down constraints (Section 5.2) and bottom-up constraints (Section 5.3). These dual perspectives are ultimately unified in a single framework—termed *correlated coin tossing* (Section 5.4).

5.1. Coin Tossing Scenarios

We want to focus on declarative *coin toss outcomes* (on the one hand) and imperative *coin-tossing scenarios* (on the other hand).

Definition 3. *Given a coin x_α, where $l \geq \alpha \geq 1$, we say that*

$$x_\alpha = b, \ \ \textit{with} \ \ b \in \{0,1\},$$

is a coin toss outcome.

Moreover, a coin-tossing scenario implementing the outcome $x_\alpha = 1$ *is represented by a proposition of the form*

$$\frac{\square}{\widehat{q_1}}@(i,j)_{\mathcal{B}} \quad \textit{or} \quad \frac{\#}{\widehat{q_1}}@(i,\widehat{c})_{\mathcal{B}},$$

where i and j denote suitable row and column indices, respectively.

Similarly, a coin-tossing scenario implementing the outcome $x_\alpha = 0$ *is represented by a proposition of the form*

$$\frac{0}{\widehat{Q_0}}@(i,j)_{\mathcal{B}},$$

where i and j denote suitable row and column indices, respectively.

The two instances of "suitable" in Definition 3 can be formalized; however, we choose to illustrate the concept through a series of examples instead. In the following examples, note that coin x_1 is the last coin tossed, as the coins are ordered as follows:

$$x_l, x_{l-1}, \ldots, x_2, x_1.$$

Example 1. *Consider $l = 4$ and Table 10. The coin toss outcome $x_4 = 1$ is exhaustively implemented by the coin-tossing scenario*

$$\frac{\square}{\widehat{q_1}}@(2,3)_{\mathcal{B}}.$$

The TM N^* can produce a result of 1 for the leftmost coin, x_4, in exactly one coin-tossing scenario.

Example 2. *Consider $l = 3$ and Table 11. The coin toss outcome $x_3 = 1$ is exhaustively implemented by the same coin-tossing scenario as in the previous example.*

Remark 12. *To facilitate Definition 3, observe that for a given coin x_α and appropriately chosen indices i and j, the conjunction*

$$\neg\frac{0}{\widehat{Q_0}}@(i+1,j-1) \ \ \wedge \ \ \frac{\square}{\widehat{q_1}}@(i,j)_{\mathcal{B}}$$

represents a coin-tossing scenario implementing the outcome $x_\alpha = 1$. Furthermore, the first conjunct trivially follows from the second when the instructions of N^*—pertaining to the Coin-Tossing Stage—are considered

 doi:10.20944/preprints202507.2367.v1

in conjunction with the Horn formula ψ^{η}_{step}. *Hence, the first conjunct is redundant and does not appear in Definition 3.*

Example 3. *Consider* $l = 4$ *and Table 10. There are four distinct coin-tossing scenarios that exhaustively implement* $x_1 = 1$*; they are:*

$$\frac{\#}{\widehat{q_1}}@(5,6)_{\mathcal{B}}, \quad \frac{\#}{\widehat{q_1}}@(7,6)_{\mathcal{B}}, \quad \frac{\#}{\widehat{q_1}}@(9,6)_{\mathcal{B}}, \quad \textit{and} \quad \frac{\#}{\widehat{q_1}}@(11,6)_{\mathcal{B}}.$$

Example 4. *Consider* $l = 3$ *and Table 11. There are three distinct coin-tossing scenarios that exhaustively implement* $x_1 = 0$*; they are:*

$$\frac{0}{\widehat{Q_0}}@(5,4)_{\mathcal{B}}, \quad \frac{0}{\widehat{Q_0}}@(7,4)_{\mathcal{B}}, \quad \textit{and} \quad \frac{0}{\widehat{Q_0}}@(9,4)_{\mathcal{B}}.$$

Example 5. *The three coin-tossing scenarios in Example 4, which involve the coin* x_1 *and* $l = 3$ *(Table 11), also exhaustively implement the coin toss outcome* $x_2 = 0$ *when* $l = 4$ *(Table 10).*

The crux in all these examples is that, for any given coin x_α and any specific coin toss outcome $x_\alpha = b$, there exists exactly one column in the basic *mini tableau* that harbors all coin-tossing scenarios implementing $x_\alpha = b$.

Definition 4. *A* composite coin-tossing scenario*, or simply a* scenario*, is a conjunction of one or more coin-tossing scenarios.*

For clarity, we introduce Example 6 prior to the definitions.

Example 6. *Consider* $l = 4$ *and Table 10. The scenario*

$$\frac{\square}{\widehat{q_1}}@(2,3)_{\mathcal{B}} \quad \wedge \quad \frac{\#}{\widehat{q_1}}@(5,6)_{\mathcal{B}}$$

implements the combined outcome

$$x_4 = 1 \ \textit{ and } \ x_1 = 1. \tag{8}$$

Definition 5. *We say that c of the form*

$$c_1 \textit{ and } \ldots \textit{ and } c_n,$$

with each c_i *of the form* $x_\alpha = b$*, is a* combined outcome *when any two conjuncts* c_i *and* c_j $(i \neq j)$ *in c pertain to a different coin.*

For instance, expression (8) represents a combined outcome; however,

$$x_4 = 1 \ \text{ and } \ x_4 = 1$$

does not, nor does

$$x_4 = 1 \ \text{ and } \ x_4 = 0.$$

Definition 6. *We say that* a scenario *s of the form*

$$s_1 \wedge \ldots \wedge s_n,$$

implements a combined outcome *c of the form*

$$c_1 \textit{ and } \ldots \textit{ and } c_n,$$

 doi:10.20944/preprints202507.2367.v1

when each conjunct s_i in s implements the corresponding conjunct c_i of c.

Lemma 1. *Given a coin x_α, with $l \geq \alpha \geq 1$, and an outcome $b \in \{0,1\}$, there are at most l distinct coin-tossing scenarios that exhaustively implement the coin toss outcome $x_\alpha = b$.*

Proof. We present a geometrical argument concerning the basic *mini tableau*, based on the definitions provided for the Coin-Tossing Stage (Section 4.1). For any non-zero value of l, no column c to the left of the rightmost column $\widehat{c}$ contains more occurrences of a dot or a hyphen. Consequently, we present an argument for the column $\widehat{c}$. (A similar argument can then be made for any other column c that contains an equal number of occurrences.)

Due to the stepwise behavior of any TM, the first l cells in column $\widehat{c}$ cannot contain a dot or a hyphen. Similarly, the last cell in that column—at column index $3l+1$—contains a hyphen (see item 1 in Section 4.1.2). Moreover, every other cell in the rightmost column, ranging from row index $l+1$ to $3l+1$, cannot contain a dot or a hyphen, owing to the properties of TM movement. Therefore, at most $\lceil x \rceil$ cells in column $\widehat{c}$ can contain a dot or a hyphen, where

$$x = \frac{(3l+1)-l}{2} = l + \frac{1}{2}.$$

This yields $l+1$. Finally at least one of these cells does not contribute to the outcome b but solely to $1-b$, establishing the desired upper bound of l. □

Corollary 1. *Consider three coins x_γ, x_β, and x_α, with $l \geq \gamma > \beta > \alpha \geq 1$, and three coin toss outcomes $b_\gamma, b_\beta, b_\alpha \in \{0,1\}$. There are at most l^3 distinct scenarios that exhaustively implement the combined outcome: $x_\gamma = b_\gamma$ and $x_\beta = b_\beta$ and $x_\alpha = b_\alpha$.*

Table 11. The basic *mini tableau*: tossing $l = 3$ coins from left to right, resulting in the truth values for x_3, x_2, and x_1. The row $\widehat{r}$ corresponds to the state attained after all coins have been tossed and when the rightmost column $\widehat{c} = 5$ is reached. In this case, l is odd. Consequently, $\widehat{r}$ is even and belongs to the set $\widehat{r} \in \{4,6,8,10\} = \{l+1, l+3, \ldots, 3l+1\}$.

1	#	—			#
2	—		—		
3		—		—	
4			—		—
5		—		·	
6			—		·
7				—	
8			—		·
9				—	
10					—
	1	2	3	4	5

Definition 7. *Consider a coin toss outcome $x_\alpha = b$, i.e., with α and b fixed. We write*

$$(x_\alpha = b)@[\beta], \tag{9}$$

with the parameter β ranging from 1 to l, to denote the β-th coin-tossing scenario within a fixed, standard ordering of all coin-tossing scenarios that implement $x_\alpha = b$. If, for sufficiently large β, expression (9) does not correspond to a coin-tossing scenario, it instead serves as a placeholder for the truth value 0, meaning false.

Remark 13. *From Lemma 1, we know that each coin toss outcome can correspond to at most l distinct coin-tossing scenarios. This explains why β in Definition 7 ranges from 1 to l.*

 doi:10.20944/preprints202507.2367.v1

Example 7. *Suppose $l = 3$ and that any scenario (cf. Definitions 4 and 6) implementing the combined outcome $x_2 = 1$ and $x_1 = 0$ (cf. Definition 5) must imply the presence of bit 1 in cell$[4,2]$ in the basic tableau (Table 11). We formulate this constraint with the following Horn formula:*

$$\bigwedge_{1\leq\beta_2\leq l}\ \bigwedge_{1\leq\beta_1\leq l}\ [\,(x_2 = 1)@[\beta_2]\ \wedge\ (x_1 = 0)@[\beta_1]\ \ \rightarrow\ \ 1@(4,2)_{\mathcal{B}}\,], \tag{10}$$

which is of complexity $O(l^2)$.

In the remainder of this subsection, we examine Example 7 (Table 11), which we will need later. We start with the following syntactic surrogates ($\equiv$) from Example 4:

- $(x_1 = 0)@[1] \quad \equiv \quad \frac{0}{\widehat{Q_0}}@(5,4)_{\mathcal{B}}$,
- $(x_1 = 0)@[2] \quad \equiv \quad \frac{0}{\widehat{Q_0}}@(7,4)_{\mathcal{B}}$,
- $(x_1 = 0)@[3] \quad \equiv \quad \frac{0}{\widehat{Q_0}}@(9,4)_{\mathcal{B}}$.

Likewise, the reader can verify that coin x_2, with $l = 3$, can land on 1 in either of the following two tossing scenarios:

- $(x_2 = 1)@[1] \quad \equiv \quad \frac{\square}{\widehat{q_1}}@(3,4)_{\mathcal{B}}$,
- $(x_2 = 1)@[2] \quad \equiv \quad \frac{\square}{\widehat{q_1}}@(5,4)_{\mathcal{B}}$.

Hence, formula (10) represents $2 \times 3 = 6$ implications of the three-literal form:

$$(x_2 = 1)@[\beta_2]\ \wedge\ (x_1 = 0)@[\beta_1]\ \ \rightarrow\ \ 1@(4,2)_{\mathcal{B}}, \tag{11}$$

as partially illustrated with the two *mini tableaux* in Table 12.

The left illustration in Table 12 depicts the two possible choices for the left conjunct in formula (11), while the right illustration presents the three possible choices for the right conjunct.

Among these six implications, only two remain potentially satisfiable when incorporating the constraints of ψ^{η}_{step} (which accounts for the step-by-step behavior of N^* on w) and ψ^{1}_{extra} (which captures the spatial dynamics of the TM's head within the tableau). These two remaining options are depicted in Table 13, with one option illustrated on the left and the other on the right. In both cases, the consequent

$$1@(4,2)_{\mathcal{B}}$$

from formula (11) is emphasized by a boldfaced 1 in the corresponding cell.

How do we transition from the left illustration (respectively the right illustration) in Table 13 to the corresponding left (respectively right) illustration in Table 14? As before, the transformation is achieved by incorporating the constraints ψ^{η}_{step} and ψ^{1}_{extra}. Moreover, only the left illustration in Table 14 remains potentially satisfiable, as the right illustration is rendered unsatisfiable due to the constraints imposed by the instructions $\widehat{t_2}$–$\widehat{t_6}$ in Section 4.1. Specifically, in the right illustration of Table 14, *cell*$[3,2]$ must contain the tape-state symbol $\frac{0}{\widehat{Q_0}}$. However, this conflicts with $1@(4,2)_{\mathcal{B}}$ and instruction $\widehat{t_6}$.

Finally, the left illustration in Table 14 can be further refined by extending the column of 1s downward:

$$1@(7,2)_{\mathcal{B}}\ \ \wedge\ \ 1@(8,2)_{\mathcal{B}}.$$

This extension follows from ψ^{1}_{extra} and the fact that row $\widehat{r}$ must have been established as $\widehat{r} = 6$ at the outset of this discussion—for else *cell*$[5,4]$ may not contain $(x_1 = 0)@[1]$. Consequently,

$$\frac{\#}{\widehat{q_1}}@(6,5)_{\mathcal{B}}$$

holds as a propositional fact.

 doi:10.20944/preprints202507.2367.v1

Table 12. Illustrating $(x_2 = 1)@[\beta_2]$ on the left and $(x_1 = 0)@[\beta_1]$ on the right.

1	#	–			#
2	–		–		
3		–		$(x_2 = 1)@[1]$	
4			–		–
5		–		$(x_2 = 1)@[2]$	
6			–		·
7				–	
8			–		·
9				–	
10					–
11	#				#
	1	2	3	4	5

1	#	–			#
2	–		–		
3		–		–	
4			–		–
5		–		$(x_1 = 0)@[1]$	
6			–		·
7				$(x_1 = 0)@[2]$	
8			–		·
9				$(x_1 = 0)@[3]$	
10					–
11	#				#
	1	2	3	4	5

Table 13. Two distinct computations in the making: one on the left and the other on the right.

1	#	–			#
2	–		–		
3		–		$(x_2 = 1)@[1]$	
4		**1**	–		–
5				$(x_1 = 0)@[1]$	
6			–		·
7				–	
8			–		·
9				–	
10					–
11	#				#
	1	2	3	4	5

1	#	–			#
2	–		–		
3		–		–	
4		**1**	–		–
5				$(x_2 = 1)@[2]$	
6			–		·
7				$(x_1 = 0)@[2]$	
8			–		·
9				–	
10					–
11	#				#
	1	2	3	4	5

Table 14. Two distinct computations taking shape, with only the left one remaining potentially satisfiable.

1	#	–			#
2	–	1	–		
3		1		$(x_2 = 1)@[1]$	
4		**1**	–		–
5		1		$(x_1 = 0)@[1]$	
6		1	–		·
7				–	
8			–		·
9				–	
10					–
11	#				#
	1	2	3	4	5

1	#	–			#
2	–		–		
3		–		–	
4		**1**	–		–
5		1		$(x_2 = 1)@[2]$	
6		1	–		·
7		1		$(x_1 = 0)@[2]$	
8		1	–		·
9				–	
10					–
11	#				#
	1	2	3	4	5

5.2. Illustrating Top-Down (↓) Constraints

A central tenet in this paper is that any particular coin-tossing scenario in the basic *mini tableau* must determine the presence of specific tape-state symbols in lower cells, in the remainder of the basic tableau.

To illustrate, consider the coin x_2 with $l = 3$ (Table 11) which can land on 1 in either of the following two coin-tossing scenarios:

$$\frac{\square}{\widehat{q_1}}@(3,4)_{\mathcal{B}} \quad \vee \quad \frac{\square}{\widehat{q_1}}@(5,4)_{\mathcal{B}},$$

as depicted in the left illustration of Table 12.

Each of these two coin-tossing scenarios necessitates the presence of tape-state symbols $\frac{s}{q}$ in cells located beyond the basic *mini tableau* but (obviously) still within the bounds of the basic tableau. This feature holds for both the Updating Stage $\overleftrightarrow{S}$ and the Checking Stage $\overleftarrow{S}$ (Section 4), giving rise to $\overleftrightarrow{\downarrow}$ constraints (Section 5.2.1) and $\overleftarrow{\downarrow}$ constraints (Section 5.2.2), respectively.

 doi:10.20944/preprints202507.2367.v1

Remark 14. *We use the notation*

$$l \rightarrow c \leftarrow r$$

as a shorthand for the conjunction

$$(l \rightarrow c) \wedge (r \rightarrow c).$$

5.2.1. Updating Stage

Recall from Section 4.2.4 that the Updating Stage $\overleftrightarrow{S}$ for coin x_2 begins with one of the following two types of configurations:

$$\&\,\$\,cl_m\,\$\ldots\$\,cl_2\,\$\,a\,a\,0 \vee \bar{a}\,1 \vee 0 \vee \$\#10\ldots\underset{\overleftrightarrow{q_2}}{1}\#\#.$$

$$\&\,\$\,cl_m\,\$\ldots\$\,cl_2\,\$\,a\,a\,0 \vee \bar{a}\,1 \vee 0 \vee \$\#10\ldots\underset{\overleftrightarrow{q_2}}{0}\#\#.$$

The first configuration type necessitates familiarity with the instructions $\overleftrightarrow{\tau_9}$ and $\overleftrightarrow{\tau_{10}}$,

$$\overleftrightarrow{\tau_9} : \left(\overleftrightarrow{U_1}, a\right) \rightarrow \left(\overleftrightarrow{Q_1}, 1, -\right), \qquad \overleftrightarrow{\tau_{10}} : \left(\overleftrightarrow{U_1}, \bar{a}\right) \rightarrow \left(\overleftrightarrow{Q_1}, 0, -\right),$$

introduced in Remark 10. The second configuration type relies on:

$$\overleftrightarrow{t_9} : \left(\overleftrightarrow{U_0}, a\right) \rightarrow \left(\overleftrightarrow{Q_0}, 0, -\right), \qquad \overleftrightarrow{t_{10}} : \left(\overleftrightarrow{U_0}, \bar{a}\right) \rightarrow \left(\overleftrightarrow{Q_0}, 1, -\right).$$

Eeach coin-tossing scenario implementing $x_2 = 1$ (high up in the basic tableau) necessarily involves either $\underset{\overleftrightarrow{U_1}}{a}$ in cell ${}_*c$ or $\underset{\overleftrightarrow{U_1}}{\bar{a}}$ in cell c_* (further down in the basic tableau), corresponding to $\overleftrightarrow{\tau_9}$ and $\overleftrightarrow{\tau_{10}}$, respectively. These cases are examined in the first and second subsections, respectively. The complexity is further analyzed in a third subsection.

Case *a*

Definition 8. *Consider a coin x_α, with $l \geq \alpha \geq 1$. An (uninstantiated, encoded) literal in $\widetilde{w}$ that contains at least α occurrences of the symbol a is called an* a-admissible literal relative to x_α.

Definition 9. *Consider a coin x_α, with $l \geq \alpha \geq 1$ along with an a-admissible literal relative to x_α, denoted as ${}_*l$. Counting from right to left, focus on the α-th occurrence of a in ${}_*l$. In the basic tableau, the* a-updating cell corresponding to coin x_α and literal ${}_*l$ *is the cell ${}_*c$ that contains either*

$$\underset{\overleftrightarrow{U_1}}{a} \quad or \quad \underset{\overleftrightarrow{U_0}}{a},$$

as prescribed by instructions $\overleftrightarrow{\tau_9}$ and $\overleftrightarrow{t_9}$, respectively.

The next claim follows from the construction of N^* in Section 4:

Claim 1. *For each fixed row index $\widehat{r}$ and coin x_α, there exists a one-to-one correspondence between the literals ${}_*l$ and the cells ${}_*c$, as established in Definition 9.*

Example 8. *Referring to Table 11, let the row index $\widehat{r}$ be initially set to $\widehat{r} = 6$. Now, consider the coin toss outcome $x_2 = 1$. Then, for each a-admissible literal ${}_*l$ in $\widetilde{w}$ relative to x_2, the following $\overleftrightarrow{\downarrow}$ constraints hold:*

$$\underset{\widehat{q_1}}{\square}@(3,4)_{\mathcal{B}} \;\rightarrow\; \underset{\overleftrightarrow{U_1}}{a}@({}_*i, {}_*j)_{\mathcal{B}} \;\leftarrow\; \underset{\widehat{q_1}}{\square}@(5,4)_{\mathcal{B}}, \tag{12}$$

where $cell[{}_*i, {}_*j] = cell\ {}_*c$, and ${}_*c$ is the a-updating cell corresponding to x_2 and ${}_*l$.

 doi:10.20944/preprints202507.2367.v1

Remark 15. *To recatipulate, given that there are some number γ of a-admissible literals ${}_*l$ relative to x_2, each of the two coin-tossing scenarios that implement $x_2 = 1$ necessitates the presence of a specific tape-state symbol in the corresponding γ cells ${}_*c$ further down in the (basic) tableau.*

Case $\overline{a}$

A similar discussion regarding $x_2 = 1$ and the symbol $\overline{a}$ (rather than a) brings us to the following definitions:

Definition 10. *Consider a coin x_α, with $l \geq \alpha \geq 1$. An (uninstantiated, encoded) literal in $\widetilde{w}$ that contains at least α occurrences of the symbol $\overline{a}$ is called an* $\overline{a}$-admissible literal relative to x_α.

Definition 11. *Consider a coin x_α, with $l \geq \alpha \geq 1$, along with an $\overline{a}$-admissible literal relative to x_α, denoted as l_*. Counting from right to left, focus on the α-th occurrence of $\overline{a}$ in l_*. In the basic tableau, the* $\overline{a}$-updating cell corresponding to coin x_α and literal l_* *is the cell c_* that contains either*

$$\overset{\overline{a}}{\underset{U_1}{\overleftrightarrow{\ }}} \quad or \quad \overset{\overline{a}}{\underset{U_0}{\overleftrightarrow{\ }}},$$

as prescribed by instructions $\overleftrightarrow{t_{10}}$ and $\overleftrightarrow{t_{10}}$, respectively.

Claim 2. *For each fixed row index $\widehat{r}$ and coin x_α, there exists a one-to-one correspondence between the literals l_* and the cells c_*, as established in Definition 11.*

Example 9. *Referring to Table 11, let the row index $\widehat{r}$ be initially set to $\widehat{r} = 6$. Now, consider the coin toss outcome $x_2 = 1$. Then, for each $\overline{a}$-admissible literal l_* in $\widetilde{w}$ relative to x_2, the following $\overleftrightarrow{\downarrow}$ constraints hold:*

$$\overset{\square}{\underset{\widehat{q}_1}{\overleftrightarrow{\ }}}@(3,4)_{\mathcal{B}} \;\rightarrow\; \overset{\overline{a}}{\underset{U_1}{\overleftrightarrow{\ }}}@(i_*, j_*)_{\mathcal{B}} \;\leftarrow\; \overset{\square}{\underset{\widehat{q}_1}{\overleftrightarrow{\ }}}@(5,4)_{\mathcal{B}}, \tag{13}$$

where cell$[i_*, j_*]$ = cell c_*, and c_* is the $\overline{a}$-updating cell corresponding to x_2 and l_*.

Remark 16. *To recatipulate, given that there are some number γ of $\overline{a}$-admissible literals l_* relative to x_2, each of the two coin-tossing scenarios that implement $x_2 = 1$ necessitates the presence of a specific tape-state symbol in the corresponding γ cells c_* further down in the (basic) tableau.*

Complexity

Generalizing from the Horn constraints (12) and (13), we take a conservative approach and assume that updating $\widetilde{w}$ to reflect a coin toss outcome $x_\alpha = b$ requires modifying all $3m$ encoded literals. In other words, this update necessitates performing $3m$ overwrites at the α-th occurrences (counting from the right) of either a or $\overline{a}$. Extending this to all l coins, and applying Lemma 1, we account for l coin-tossing scenarios per coin outcome, with two possible outcomes per coin. Consequently, the total number of $\overleftrightarrow{\downarrow}$ constraints, each expressed in the two-literal Horn form

$$\ldots@\ldots \rightarrow \ldots@\ldots,$$

is given by

$$3m \times l \times l \times 2 \quad = \quad O(n^3).$$

Remark 17. *The complexity expressed in terms of l, with $m = O(l^3)$, is $O(l^5)$.*

Additional $\overleftrightarrow{\downarrow}$ constraints can be introduced, yet still resulting in merely $O(n^\kappa)$ complexity, for some constant κ. Another example is provided in Appendix C. To establish our compression result (Theorem 2), it is not necessary to exhaustively list all $\overleftrightarrow{\downarrow}$ constraints.

 doi:10.20944/preprints202507.2367.v1

5.2.2. Checking Stage

Concerning the Checking Stage $\overleftarrow{S}$, we wish to give examples of $\overleftarrow{\downarrow}$ constraints related to the outcome $x_2 = 1$. However, this depends on the positioning of both x_2 and $\neg x_2$ within ϕ and, in its encoded form, within $\widetilde{w}$. Hence, for illustration, suppose that the formula ϕ contains x_2 only as the leftmost literal encoded in $\widetilde{w}$, and $\neg x_2$ only as the rightmost literal. Under this assumption, the uninstantiated word w takes the form:

$$\&\,\$\,a\,a\,\vee\ldots\,\$\ldots\$\,cl_2\,\$\,\ldots\,\vee\,\overline{a}\,\overline{a}\,\vee\,\$\#\ldots\#.$$

Now, recall from Section 4.3 the following three instructions:

$$\overleftarrow{t_5}:\left(\overleftarrow{Q_0},\vee\right)\rightarrow\left(\overleftarrow{R_0},\vee,+\right),$$

$$\overleftarrow{t_{15}}:\left(\overleftarrow{Q_{000}},\$\right)\rightarrow\left(\overleftarrow{R_{000}},\$,+\right),$$

$$\overleftarrow{t'_{15}}:\left(\overleftarrow{S_{000}},\$\right)\rightarrow\left(\overleftarrow{T_{000}},\$,+\right).$$

These correspond to the following three kinds of configurations, respectively:

$$\&\,\$\,1\,0\,\vee\ldots\,\$\ldots\$\,cl_2\,\$\,\ldots\,\vee\,\underset{\overleftarrow{R_0}}{0}\,1\,\vee\,\$\#\ldots\#,$$

$$\&\,\$\,\underset{\overleftarrow{R_{000}}}{1}\,0\,\vee\ldots\,\$\ldots\$\,cl_2\,\$\,\ldots\,\vee\,0\,1\,\vee\,\$\#\ldots\#,$$

$$\&\,\$\,\underset{\overleftarrow{T_{000}}}{1}\,0\,\vee\ldots\,\$\ldots\$\,cl_2\,\$\,\ldots\,\vee\,0\,1\,\vee\,\$\#\ldots\#.$$

Focusing on the rightmost and leftmost encoded literals (cf. $\overleftarrow{R_0}$ on the one hand, and $\overleftarrow{R_{000}}$ and $\overleftarrow{T_{000}}$ on the other hand) we observe that two corresponding cells, c_* and ${}_*c$, contain 0 and 1, respectively:

$$\underset{\widehat{q_1}}{\square}@(3,4)_{\mathcal{B}}\ \rightarrow\ \underset{\overleftarrow{R_0}}{0}@(i_*,\,j_*)_{\mathcal{B}}\ \leftarrow\ \underset{\widehat{q_1}}{\square}@(5,4)_{\mathcal{B}},\tag{14}$$

$$\left[\underset{\widehat{q_1}}{\square}@(3,4)_{\mathcal{B}}\ \wedge\ \underset{\overleftarrow{Q_{000}}}{\$}@({}_*i-1,\,{}_*j-1)_{\mathcal{B}}\right]\ \rightarrow\ \underset{\overleftarrow{R_{000}}}{1}@({}_*i,\,{}_*j)_{\mathcal{B}},\tag{15}$$

$$\left[\underset{\widehat{q_1}}{\square}@(3,4)_{\mathcal{B}}\ \wedge\ \underset{\overleftarrow{S_{000}}}{\$}@({}_*i-1,\,{}_*j-1)_{\mathcal{B}}\right]\ \rightarrow\ \underset{\overleftarrow{T_{000}}}{1}@({}_*i,\,{}_*j)_{\mathcal{B}}.\tag{16}$$

Three points warrant clarification. First, the cells c_* and ${}_*c$ "contain 0 and 1, respectively," as indicated by the use of tape-state symbols: $\underset{\overleftarrow{R_0}}{0}$ on the one hand, and $\underset{\overleftarrow{R_{000}}}{1}$ along with $\underset{\overleftarrow{T_{000}}}{1}$ on the other hand. Second, cells c_* and ${}_*c$ are shorthand for $cell[i_*,\,j_*]$ and $cell[{}_*i,\,{}_*j]$, respectively. Third, the reader will recognize that the counterparts to formulas (15) and (16), where the left conjunct is replaced by

$$\underset{\widehat{q_1}}{\square}@(5,4)_{\mathcal{B}},$$

should also be considered in our discussion. For the sake of brevity, we omit them.

Complexity

The crux is again that every coin-tossing scenario implementing $x_2 = 1$ (positioned high up in the basic tableau) necessitates the presence of specific tape-state symbols appearing further down in the basic tableau. However, now we have also encountered three-literal Horn formulas (e.g., formulas (15) and (16)) instead of only two-literal Horn formulas (such as formula (14)).

 doi:10.20944/preprints202507.2367.v1

Generalizing from the specific constraints illustrated in (14)–(16), we assume that checking—during the Checking Stage $\overleftarrow{S}$—whether the instantiated $\widetilde{w}$ is trivially true or trivially false requires examining each leftmost bit in all $3m$ encoded and instantiated literals. Given that there are l coins, at most l coin-tossing scenarios per coin outcome, and two possible outcomes per coin toss, this results in a complexity of

$$3m \times l \times l \times 2 \quad = \quad O(n^3)$$

constraints, each expressed in either two-literal Horn form,

$$\ldots @ \ldots \rightarrow \ldots @ \ldots,$$

or three-literal Horn form:

$$\ldots @ \ldots \wedge \ldots @ \ldots \rightarrow \ldots @ \ldots .$$

Remark 18. *Formula* (15)*—and many other formulas—can be extended as follows:*

$$\left[\frac{\#}{\widehat{q}_1} @(\widehat{r}, \widehat{c})_{\mathcal{B}} \quad \wedge \quad \frac{\square}{\widehat{q}_1} @(3,4)_{\mathcal{B}} \quad \wedge \quad \frac{\$}{\overleftarrow{Q_{000}}} @({}_*i-1, {}_*j-1)_{\mathcal{B}} \right] \rightarrow \frac{1}{\overleftarrow{R_{000}}} @({}_*i, {}_*j)_{\mathcal{B}} \tag{17}$$

This more elaborate formulation also aligns with our discourse, in which the guess

$$\frac{\#}{\widehat{q}_1} @(\widehat{r}, \widehat{c})_{\mathcal{B}},$$

is explicitly part of the equation. Nevertheless, we continue to assume that the Horn constraints—such as (15)*—are generated dynamically, thus rendering the first conjunct in formula* (17) *redundant.*

5.3. Illustrating Bottom-Up (↑) Constraints

Bottom-up constraints are not a new consideration in this discussion. In fact, a specific type of bottom-up constraint has already emerged in the transition from the left illustration in Table 13 to the left illustration in Table 14. This transition showcases the upward propagation of 1s, driven by ψ^1_{extra}. However, another form of bottom-up constraint (↑) also merits attention, which we will briefly explore here.

We revisit Clause 1 from Section 4.2.1, along with its unary encoding, cl_1:

$$a\, a\, a \vee \overline{a}\, \overline{a} \vee a \vee .$$

Assuming this encoded clause is integral to $\widetilde{w}$, let $x_2 = 1$ and $x_1 = 0$.

During the Updating Stage $\overleftrightarrow{S}$ of the machine N^*, the coresponding string encoding ultimately takes one of the following forms:

$$1\,1\,0 \vee 0\,1 \vee 0 \vee,$$

or the form

$$0\,1\,0 \vee 0\,1 \vee 0 \vee,$$

depending on whether $x_3 = 1$ or, respectively, $x_3 = 0$.

As established in Section 4.3.4, the leftmost bit in these forms corresponds respectively to the following instructions:

$$\overleftarrow{t_{16}} : \left(\overleftarrow{R_{000}}, 1\right) \rightarrow \left(\overleftarrow{q_0}, 1, -\right), \qquad \overleftarrow{t_{17}} : \left(\overleftarrow{R_{000}}, 0\right) \rightarrow \left(q_{reject}, 0, -\right).$$

The first form alone does not result in the unsatisfiability of

$$\psi_{trim} \wedge \ldots,$$

 doi:10.20944/preprints202507.2367.v1

where the ellipsis (...) represents, among other factors, the guessed coin-tossing scenarios that implement $x_2 = 1$ and $x_1 = 0$. In contrast, the second form does lead to unsatisfiability due to the q_{reject} state symbol and the ψ_{accept} constraint (recall Section 4.3.6).

Hence, we contend that a particular kind of bottom-up constraint ($\uparrow$) is at play, extending from a lower cell $\underline{c}$ in the basic tableau—which, in adherence to $\overleftarrow{t_{16}}$ and $\overleftarrow{t_{17}}$, potentially contains the $\frac{1}{\overleftarrow{R_{000}}}$ or $\frac{0}{\overleftarrow{R_{000}}}$ symbol—to the truth value b of x_3:

$$b@(4,2)_{\mathcal{B}}, \tag{18}$$

as determined higher up, in the basic *mini tableau* (see the left illustration in Table 13).

A key consideration is that cell $\underline{c}$ may represent a hole in the basic tableau, preventing any symbol—including $\frac{1}{\overleftarrow{R_{000}}}$ and $\frac{0}{\overleftarrow{R_{000}}}$—from being turned on. Now, by automatically filling it with $\frac{1}{\overleftarrow{R_{000}}}$ instead of $\frac{0}{\overleftarrow{R_{000}}}$, the hole is resolved without user intervention. However, this does require an extra measure beyond top-down reasoning, leading us to the topic of *correlated coin-tossing constraints*.

5.4. Synthesis: Correlated Coin Tossing Constraints

We integrate the top-down and bottom-up perspectives on the `3-SAT` solver N^* within a framework that focuses exclusively on the *mini tableau*. To achieve this, let us contemplate an arbitrary clause in ϕ, which is typically not a Horn formula.

Consider for instance Clause 1:

$$x_3 \vee \overline{x_2} \vee x_1.$$

Any of the following three equivalent formulations of Clause 1—where each antecedent represents a combined toss outcome of two coins (of the three coins in question)—are also, quite evidently, non-Horn formulas:

$$x_2 \wedge \overline{x_1} \rightarrow x_3 \tag{19}$$

$$\overline{x_3} \wedge \overline{x_1} \rightarrow \overline{x_2} \tag{20}$$

$$\overline{x_3} \wedge x_2 \rightarrow x_1 \tag{21}$$

Yet, by synthesizing insights from both top-down and bottom-up reasoning, we can reformulate each of these expressions ((19)–(21)) as a compact family of Horn clauses.

Without loss of generality, we focus on implication (19) in this section. On the one hand, we express the consequent of (19) in the form of proposition (18) with $b = 1$,

$$1@(4,2)_{\mathcal{B}},$$

thereby establishing the coin toss outcome $x_3 = 1$ (Table 13). On the other hand, we express the antecedent of (19) as a scenario (see Definition 4) that implements the combined outcome

$$x_2 = 1 \ \text{ and } \ x_1 = 0.$$

Naturally, we must account for all possible such scenarios:

$$\bigwedge_{1 \le \beta_2 \le l} \ \bigwedge_{1 \le \beta_1 \le l} [\,(x_2 = 1)@[\beta_2] \ \wedge \ (x_1 = 0)@[\beta_1] \ \rightarrow \ 1@(4,2)_{\mathcal{B}}\,], \tag{22}$$

which is Horn formula (10) from Example 7.

We have transformed the non-Horn formula (19) into the Horn formula (22). Similarly, the Horn formulas for implications (20) and (21) are, respectively:

$$\bigwedge_{1 \le \beta_3 \le l} \ \bigwedge_{1 \le \beta_1 \le l} [\,(x_3 = 0)@[\beta_3] \ \wedge \ (x_1 = 0)@[\beta_1] \ \rightarrow \ 0@(5,3)_{\mathcal{B}}\,], \tag{23}$$

 doi:10.20944/preprints202507.2367.v1

$$\bigwedge_{1\le\beta_3\le l}\ \bigwedge_{1\le\beta_2\le l}\ [\,(x_3=0)@[\beta_3]\ \wedge\ (x_2=1)@[\beta_2]\ \ \rightarrow\ \ 1@(8,4)_{\mathcal{B}}\,]. \tag{24}$$

To summarize, the coins x_3, x_2, and x_1 are correlated via Clause 1. Initially, this correlation is expressed through non-Horn constraints (19)–(21) and, ultimately, through Horn constraints (22)–(24). Importantly, our method applies to every 3-literal clause in ϕ.

Complexity

Each correlated coin-tossing constraint (e.g. (22)) consists of $O(l^2)$ literals. Moreover, there are three such constraints per clause in ϕ (cf. (22)–(24) for Clause 1), with a total of m clauses. Consequently, the overall complexity amounts to:

$$O(l^2)\times 3m \ \ =\ \ O(n^3)$$

literals.

Remark 19. *The complexity expressed in terms of l, with $m = O(l^3)$, is $O(l^5)$.*

By specifying all correlated coin-tossing constraints related to ϕ, we can eliminate the $\overleftrightarrow{\downarrow}$ constraints (Section 5.2.1) and $\overleftarrow{\downarrow}$ constraints (Section 5.2.2), rendering the operation of N^* beyond the *mini tableau* obsolete. While the Coin-Tossing Stage $\widehat{S}$ remains relevant, the specifics of the Updating Stage $\overleftrightarrow{S}$ and Checking Stage $\overleftarrow{S}$ are no longer necessary. This improvement leads from the `rFHB` algorithm to the streamlined variant called `sFHB`.

6. Two Algorithms: `rFHB` and `sFHB`

We now introduce and analyze the `rFHB` and `sFHB` algorithms. Both are governed by the Horn formula ψ_{trim}, defined as:

$$\psi_{trim} \ \ =\ \ \left(\psi^{\eta}_{step}\wedge\psi_{start}\wedge\psi_{accept}\wedge\psi'_{cell}\wedge\psi^{1}_{extra}\right)\ \wedge\ \psi^{2}_{extra}. \tag{25}$$

The final conjunct, ψ^2_{extra}, varies considerably, depending on the algorithm:

- ψ^2_{extra}[`rFHB`] includes the $\overleftrightarrow{\downarrow}$ and $\overleftarrow{\downarrow}$ constraints (Section 5.2),
- ψ^2_{extra}[`sFHB`] captures the correlated coin-tossing constraints (Section 5.4).

We present *claims*, statements that we assert to be true based on the constructive nature of our chain of reasoning. Claim 3, for instance, has already been established.

Claim 3. *In either case of* `rFHB` *and* `sFHB`, *ψ_{trim} is $O(n^{\kappa_1})$ literals long, for some constant κ_1.*

The `rFHB` operates over the entire tableau, whereas `sFHB` is confined to the *mini tableau*. In Section 6.1, we describe `rFHB` with respect to the `3-SAT` solver N^*, i.e., with a fixed value of $k = 2$ and unary encoding. We also consider the alternative scenario where $k = 1$ and binary notation is used. In Section 6.2, by contrast, `sFHB` is defined independently of both k and the encoding scheme for literals, because it restricts itself to the Coin-Tossing Stage.

6.1. The `rFHB` *Algorithm*

The `rFHB` algorithm leverages the internal mechanics of the `3-SAT` solver in question. We define `rFHB` in Section 6.1.1, and provide detailed explanations of steps 1 and 3 in Section 6.1.2 and Section 6.1.3, respectively. A comprehensive cost analysis is presented in Section 6.1.4.

6.1.1. Boxed Definition

To appreciate the specifics of the `rFHB` algorithm, recall the following points:

 doi:10.20944/preprints202507.2367.v1

- Row index $\widehat{r}$ in the basic *mini tableau* corresponds to row index $3\widehat{r}-2$ in the extended *mini tableau* (Section 3).
- If and when all l coins have been tossed, the entire (extended) tableau is determined.
- The original FHB algorithm, denoted as $\mathcal{A}$, relies on a HORNSAT solver $\mathcal{H}$ (Section 3.4).
- The inferences from tape-state symbols within the *mini tableau* to those outside of it must be preprogrammed. This corresponds to formally defining $\psi^2_{extra}[\texttt{rFHB}]$.

The rFHB Algorithm Applied to N^* and w

The 3-SAT solver N^* runs in $O(n^2)$ nondeterministic time on input w of length n. Consider the corresponding extended $(3n^2+1)\times(n^2+2)$ tableau. Let l denote the number of distinct (encoded) propositional variables in w. Consider the corresponding basic $(3l+1)\times(3l+3)$ tableau $\mathcal{B}$, along with the extended $(3\cdot(3l+1)+1)\times(3l+3)$ tableau $\mathcal{T}$. We refine the original FHB algorithm $\mathcal{A}$ using top-down reasoning, captured with $\psi^2_{extra}[\texttt{rFHB}]$, as follows:

1. Guess a row $\widehat{r}$ such that $\frac{\#}{\widehat{q}_1}@(\widehat{r},\widehat{c})_{\mathcal{B}}$ is fixed,
   - $l+1\leq\widehat{r}\leq 3l+1$,
   - l is even iff $\widehat{r}$ is odd,

   where $\widehat{c}$ is the index of the rightmost column (e.g., $\widehat{c}=6$ in Table 10). Restrict attention to rows $r\in\{1,\ldots,\widehat{r}\}$ of the *mini tableau* $\mathcal{B}$, denoted $\mathcal{B}_{\widehat{r}}$, until backtracking from the guessed $\widehat{r}$.
2. Restrict $\mathcal{A}$ to only make guesses within $\mathcal{T}_{\widehat{r}}$, the extended tableau induced by $\mathcal{B}_{\widehat{r}}$, in adherence to the $\widehat{r}$-guess. The algorithm proceeds rowwise, starting with the central row $\lceil\frac{\widehat{r}}{2}\rceil$. Subsequently, in stage two, the rows $\lceil\frac{\widehat{r}}{4}\rceil$ and $\lceil\frac{3\widehat{r}}{4}\rceil$ are visited. In stage three, the rows $\lceil\frac{\widehat{r}}{8}\rceil$, $\lceil\frac{3\widehat{r}}{8}\rceil$, $\lceil\frac{5\widehat{r}}{8}\rceil$, and $\lceil\frac{7\widehat{r}}{8}\rceil$ are visited. And so on.
3. For each guessed $\frac{s^*}{q^*}@(i^*,j^*)$ by $\mathcal{A}$ within $\mathcal{T}_{\widehat{r}}$, invoke the solver $\mathcal{H}$ on

$$\psi_{trim}\wedge\ldots\wedge\frac{s^*}{q^*}@(i^*,j^*),$$

   where the ellipsis represents all other guesses under consideration. The solver specifically enforces $\psi^2_{extra}[\texttt{rFHB}]$, thereby injecting the corresponding tape-state symbols beyond the boundaries of $\mathcal{T}_{\widehat{r}}$—though naturally still confined within the entire tableau.
4. Backtracking from $\frac{s^*}{q^*}@(i^*,j^*)$ by $\mathcal{A}$ involves backtracking of the corresponding symbols injected both outside and inside $\mathcal{T}_{\widehat{r}}$.

6.1.2. Elaborating Step 1

The $\widehat{r}$-guess introduced in step 1 of the boxed definition conceptually implies that several cells beyond $\mathcal{B}_{\widehat{r}}$ can be immediately crossed out (see item $2(b)$ in Section 4.1.2). Conceptually again, these cell crossings are revoked when $\mathcal{A}$ backtracks from the $\widehat{r}$-guess. In reality, however, $\mathcal{A}$ does not work with crosses, nor does the solver $\mathcal{H}$ actually inject symbols in an actual tableau.

Recall Remark 18, where we noted that numerous formulas admit extensions. For example, formula (26) can be extended into the form of (27), shown below:

$$\left[\frac{\square}{\widehat{q}_1}@(3,4)_{\mathcal{B}}\ \wedge\ \frac{\$}{\overleftarrow{Q_{000}}}@({}_*i-1,\,{}_*j-1)_{\mathcal{B}}\right]\ \rightarrow\ \frac{1}{\overleftarrow{R_{000}}}@({}_*i,\,{}_*j)_{\mathcal{B}},\tag{26}$$

$$\left[\frac{\#}{\widehat{q}_1}@(\widehat{r},\widehat{c})_{\mathcal{B}}\ \wedge\ \frac{\square}{\widehat{q}_1}@(3,4)_{\mathcal{B}}\ \wedge\ \frac{\$}{\overleftarrow{Q_{000}}}@({}_*i-1,\,{}_*j-1)_{\mathcal{B}}\right]\ \rightarrow\ \frac{1}{\overleftarrow{R_{000}}}@({}_*i,\,{}_*j)_{\mathcal{B}}\tag{27}$$

Formula (27) makes the $\widehat{r}$-guess explicit—through its first conjunct. However, we opt for the dynamic generation of (26) and similar constraints, rendering the first conjunct in (27) redundant.

 doi:10.20944/preprints202507.2367.v1

Claim 4. *The dynamic generation of constraint* (26)*, along with all other Horn constraints encapsulated by* ψ_{trim}*, can be performed in* $O(n^{\kappa})$ *time for some sufficiently large constant* κ*.*

Claim 4 follows almost directly from Claim 3. To see this, first consider storing all constraints in a database using the format specified in (27) (after all). Then, when the $\widehat{r}$-guess is made at runtime, prune the relevant constraints from the database and discard the first conjunct (which encodes the $\widehat{r}$-guess) before passing them to $\mathcal{H}$.

6.1.3. Elaborating Step 3

Recall Example 8 with regard to Table 11, row index $\widehat{r} = 6$, and the coin toss outcome $x_2 = 1$. Then, for each a-admissible literal ${}_*l$ in $\widetilde{w}$ relative to x_2, the following two $\overset{\leftrightarrow}{\downarrow}$ constraints must hold:

$$\underset{\widehat{q}_1}{\square}@(3,4)_{\mathcal{B}} \;\rightarrow\; \overset{a}{\underset{U_1}{\Leftrightarrow}}@({}_*i,\,{}_*j)_{\mathcal{B}} \;\leftarrow\; \underset{\widehat{q}_1}{\square}@(5,4)_{\mathcal{B}}, \tag{28}$$

where $cell[{}_*i,\,{}_*j]$ is the a-updating cell corresponding to x_2 and ${}_*l$.

Now, if in step 3 the tape-state symbol $\underset{\widehat{q}_1}{\square}$ is guessed for $cell[3,4]$ in the basic *mini tableau*,

$$\text{i.e.,}\quad \underset{\widehat{q}_1}{\square}@(3,4)_{\mathcal{B}} \quad \text{or, equivalently,} \quad \underset{\widehat{q}_1}{\square}@(3\cdot 3-2,4),$$

then—by virtue of the left implication in (28)—the solver $\mathcal{H}$ must incorporate the consequent $\overset{a}{\underset{U_1}{\Leftrightarrow}}@({}_*i,\,{}_*j)_{\mathcal{B}}$ as a propositional fact in its subsequent satisfiability analysis.

Conceptually, the solver $\mathcal{H}$ injects the tape-state symbol $\overset{a}{\underset{U_1}{\Leftrightarrow}}$ beyond the boundaries of $\mathcal{T}$; specifically, in $cell[{}_*i,\,{}_*j]$ in the basic tableau. A parallel consideration applies to $cell[5,4]$ in relation to the right implication in (28).

6.1.4. Complexity

Concerning the complexity of the `rFHB` algorithm, we begin with three observations. **Observation 1** comprises three points:

1. The dimensions of the basic *mini tableau* are given by

$$O(l) \times O(l) = O(l^2). \tag{29}$$

The same result holds for the extended *mini tableau*.

2. The `rFHB` algorithm runs on N^*, implying $k = 2$ and the use of unary notation. Asymptotically, this leads to $O(l)$ symbols per literal across the total $3m$ literals in $\widetilde{w}$, forming the horizontal dimension of the entire tableau:

$$n = O(l \times m). \tag{30}$$

For the vertical dimension, we focus on the Updating Stage, which asymptotically dominates the Coin-Tossing and Checking Stages. Here, we account for l iterations over the entire length of $\widetilde{w}$:

$$l \times O(l \times m) = O(l^2 \times m). \tag{31}$$

Thus, the basic tableau, which is a quasi-square matrix, has the following dimensions:

$$O(l^2 \times m) \times O(l^2 \times m). \tag{32}$$

The same result holds for the extended tableau.

3. Comparing the overall tableau size ((32)) to the basic *mini tableau* ((29)), we obtain:

$$O(l^2 \times m^2). \tag{33}$$

 doi:10.20944/preprints202507.2367.v1

This indicates that as l (and thereby m) increases, the *mini tableau* becomes proportionally much smaller relative to the entire tableau.

Observation 2, consisting of three similar points, concerns the state of affairs had we employed a sophisticated and efficient `3-SAT` solver—with $k = 1$ and binary notation:

1. The *mini tableau* remains unchanged: see (29).
2. The horizontal dimension of the entire tableau, previously given in (30), now becomes:

$$n = O((\log l) \times m), \tag{34}$$

since only $O(\log l)$ symbols per literal are needed, in contrast to l symbols. The vertical dimension, updated from (31), becomes:

$$O(1) \times O((\log l) \times m) = O((\log l) \times m), \tag{35}$$

reflecting the fact that the Updating Stage now operates in a single (essentially right-to-left) pass. Accordingly, the tableau forms a quasi-square matrix with dimensions:

$$O((\log l) \times m) \times O((\log l) \times m) = O(\log^2 l) \times O(m^2) \tag{36}$$

3. The ratio of the entire tableau ((36)) to the *mini tableau* ((29)) is:

$$\frac{O(\log^2 l) \times O(m^2)}{O(l^2)}.$$

A devil's advocate will attempt to minimize this expression by assuming $m = \Theta(l)$, yielding:

$$\Omega(\log^2 l), \tag{37}$$

which, similar to the result in (33), still confirms that the *mini tableau* becomes proportionally smaller as l increases.

With respect to the `rFHB` algorithm, **Observation 3** underscores that Theorem 2 from Daylight [1]—which establishes genuine compression for the case $k = 1$—is derived under a notably conservative assumption: the ratio between the entire tableau and the *mini tableau* is fixed at a constant value (specifically, 2), rather than allowed to grow with l, as is the case in results (37) and (33).

Theorem 2. *(Reproduced from Daylight [1](p. 30)) Let $\langle N, k \rangle$ be a* `3-SAT` *solver. Then the runtime $R(n)$ of the* `rFHB` *algorithm pertaining to N satisfies the upper bound:*

$$R(n) \leq K^{n^{0.67k}},$$

for some constant $K > 0$.

Rewriting the upper bound from Theorem 2 in terms of l, under the assumption $m = \Theta(l)$, yields:

$$K^{n^{0.67k}} = \begin{cases} K^{O\left(l^{1.34k}\right)} & \text{if } n = O\left(l^2\right) \\ K^{O\left((l \cdot \log l)^{0.67k}\right)} & \text{if } n = O(l \cdot \log l), \end{cases}$$

where the first and second cases correspond to unary and binary notation, respectively. Recall (30) and (34), respectively.

 doi:10.20944/preprints202507.2367.v1

Corollary 2. *Let $\langle N, 1\rangle$ be a* `3-SAT` *solver, operating in binary notation. Assume $m = \Theta(l)$. Then, the runtime $R(l)$ of the* `rFHB` *algorithm associated with N admits the upper bound:*

$$R(l) \leq C^{(l \cdot \log l)^{0.67}},$$

for some constant $C > 0$.

Corollary 2 reveals genuine compression when $m = \Theta(l)$, suggesting that $\mathcal{NP}$ machines are inferior to at least one exponential time deterministic TM. Yet rather than examining this potential novelty, we stress (again) that Corollary 2 is founded on a highly conservative premise: the hole-filling region external to the *mini tableau* matches the *mini tableau* in size for all values of l.

Remark 20. *In technical terms, we highlight that Daylight's proof outline [1] (p. 31) treats the "reduction factor," denoted as Δ, as a constant—rather than as a function $\Delta(l)$ that decreases monotonically with l, such that $\lim_{l\to\infty} \Delta(l) = 0$.*

The asymptotic reality is that, as l grows, the contribution of the *mini tableau* becomes an increasingly negligible portion of the overall tableau. Recall (37) and (33). We now formalize this observation with the following theorem.

Theorem 3. *(Refinement of Theorem 2) Let $\widetilde{N}$ be some $\langle N, k\rangle$ machine that solves* `3-SAT`*, with $k \in \{1, 2\}$, working in unary or binary. Let l denote the number of distinct (encoded) propositional variables in the input w of $\widetilde{N}$. Then, the runtime $R(l)$ of the* `rFHB` *algorithm associated with $\widetilde{N}$ and w admits the upper bound:*

$$R(l) \leq (l \cdot \log l)^C, \tag{38}$$

where $C > 0$ is a constant.

Remark 21. *Theorem 3 also concerns the machine N^*, as defined in Section 4.*

Proof. We can distinguish between eight cases based on three parameters. First, we consider two values for k: either $k = 1$ or $k = 2$. Second, the machine operates using either unary or binary notation. Third, we differentiate between $m = \Theta(l)$ and $m = \Theta(l^3)$. In the remainder of this proof, it suffices to analyze the gravest case, which—as the reader can verify—occurs when $k = 1$, binary notation is used, and $m = \Theta(l)$.

In this context, recall from (37) the following result:

$$\texttt{ratio}(l) = \log^2 l,$$

which represents the smallest conceivable ratio between the size of the entire tableau and that of the *mini tableau*.

Furthermore, we reuse the recurrence relation from Daylight [1] (p. 31), which takes the following form:

$$T(p) = \kappa_0 \cdot \sqrt{p} \cdot T\left(\frac{p}{2 \cdot \texttt{ratio}(l)}\right)^2,$$

where the constant $\kappa_0 > 0$ depends on the specific TM, $\widetilde{N}$, under analysis. Appendix D presents a standard derivation of the solution to this recurrence relation, yielding:

$$T(p) = p^{O(1)}. \tag{39}$$

Additionally, as noted by Daylight [1](p. 31), at the onset of the recurrence, p denotes the initial area of possible binary nondeterministic choices made by $\widetilde{N}$. Consequently, we revisit equation (36),

 doi:10.20944/preprints202507.2367.v1

this time replacing m with l, justified by the asymptotic relationship $m = \Theta(l)$. Substituting $O(\log^2 l) \times O(l^2)$ for p in (39), we derive:

$$(l \cdot \log l)^C, \tag{40}$$

for some constant $C > 0$.

Finally, we account for the polynomial overhead per step of the rFHB algorithm, which is $O(n^\kappa)$ for some constant κ—recall Claims 3 and 4. Given that $n = O(m \cdot \log l) = O(l \cdot \log l)$, it follows that our final solution is also in the form of (40). □

In light of the implication of Theorem 3, we present an alternative approach that leads to the same conclusion: an iterative application of Theorem 2 outlined in Appendix E. We now turn to the streamlined variant of the rFHB algorithm.

6.2. *The* sFHB *Algorithm*

The sFHB focuses on the operation of the coin-tossing machine $\widehat{N}$, which is solely tasked with executing stage $\widehat{S}$ of N^* in $O(l)$ nondeterministic time (Section 4.1). Now we define:

$$\psi_{trim} \;=\; \left(\psi^{\eta}_{step} \wedge \psi_{start} \wedge \psi_{accept} \wedge \psi'_{cell} \wedge \psi^{1}_{extra}\right) \wedge \psi^{2}_{extra}[\mathtt{sFHB}], \tag{41}$$

where each component within the parentheses is of smaller size relative to the rFHB case. In particular, ψ^{η}_{step} encodes only the instructions of $\widehat{N}$, not N^*. The formula ψ_{start} characterizes the initial coin-tossing configuration (7) in Section 4.1 (without $\widetilde{w}$). Similarly, the remaining three conjuncts within the parentheses pertain solely to the structure of the (extended) *mini tableau*, not the (extended) entire tableau.

We distinguish between two arrangements that the sFHB can make per visited column (Section 6.2.1). Then we specify the algorithm (Section 6.2.2) and elaborate on step 4 in the specification (Section 6.2.3). Finally, we present a cost analysis (Section 6.2.4).

6.2.1. Two Arrangements

As with rFHB, the sFHB algorithm performs an $\widehat{r}$-guess expressed as

$$\frac{\#}{\widehat{q}_1}@(\widehat{r},\widehat{c})_{\mathcal{B}},$$

from which it can subsequently backtrack.

In contrast to rFHB, which processes the tableau rowwise, sFHB works across the columns (of the *mini tableau*). For each visited column c in the *mini tableau* (see, e.g., Table 11), sFHB guesses one of two possible symbol arrangements occurring at some position r among the first $\widehat{r}$ candidates in that column. The first possible arrangement corresponds to a coin toss yielding a 1:

$$\frac{\square}{\widehat{q}_1}@(r,c)_{\mathcal{B}} \;\wedge\; 1@(r+1,c)_{\mathcal{B}} \;\wedge\; \overline{0@(r+2,c)_{\mathcal{B}}}, \tag{42}$$

while the second corresponds to a toss yielding a 0:

$$\frac{\square}{\widehat{q}_1}@(r,c)_{\mathcal{B}} \;\wedge\; 0@(r+1,c)_{\mathcal{B}} \;\wedge\; \frac{0}{\widehat{Q}_0}@(r+2,c)_{\mathcal{B}}. \tag{43}$$

When sFHB selects an arrangement at column c and row r in the basic *mini tableau*, it introduces a *separation of concerns* between the columns to the left and those to the right of c. While subsequent hole-filling choices on either side will typically constrain the options on the other side, they will not affect the corresponding computations *an sich*. Specifically, a commitment is made—either to the first arrangement (42) or the second arrangement (43)—that constrains the tape head of $\widehat{N}$ to visit column c in the following manner:

 doi:10.20944/preprints202507.2367.v1

- in the first arrangement ((42)), solely via $cell[r,c]$ and, optionally, also via $cell[r+2,c]$;
- in the second arrangement ((43)), solely via both $cell[r,c]$ and $cell[r+2,c]$.

The sFHB algorithm populates all remaining cells in column c—namely, those above $cell[r,c]$ and below $cell[r+2,c]$, in the basic *mini tableau*—with designated tape symbols, in full compliance with the operation of the coin-tossing machine $\widehat{N}$. Specifically, blank symbols ($\square$) are inserted above row r in both arrangements, while the symbol 1 (respectively, 0) is written below row $r+2$ in the first (respectively, second) arrangement.

To illustrate the first arrangement, recall instruction $\widehat{t_1}$ (Section 4.1) and Table 11 (Section 5.1). Suppose $x_2 = 1$ and specifically:

$$\frac{\square}{\widehat{q_1}}@(2,3)_{\mathcal{B}} \;\wedge\; 1@(3,3)_{\mathcal{B}} \;\wedge\; \overline{0@(4,3)_{\mathcal{B}}}, \tag{44}$$

where the latter conjunct implicitly refers to either $1@(4,3)_{\mathcal{B}}$ or to $\frac{1}{\widehat{q_0}}@(4,3)_{\mathcal{B}}$, depending on the toss outcome for coin x_1. Regardless of which of these two possibilities manifests, the computations to the left of column 3 in Table 11 remain unaffected. This invariance is ensured by the first conjunct in (44), which enforces the separation of concerns.

We now *redefine* notation (9) from Definition 7 via two examples.

Example 10. *Based on Table 11:*

- $(x_2 = 1)@[1] \quad \equiv \quad$ *the contents of* (44)
- $(x_2 = 1)@[2] \quad \equiv \quad \frac{\square}{\widehat{q_1}}@(4,3)_{\mathcal{B}} \;\wedge\; 1@(5,3)_{\mathcal{B}} \;\wedge\; \overline{0@(6,3)_{\mathcal{B}}}$
- $(x_2 = 1)@[3] \quad \equiv \quad 0 \quad$ *(i.e.,* false*)*

Example 11. *Based on Table 11:*

- $(x_2 = 0)@[1] \quad \equiv \quad \frac{\square}{\widehat{q_1}}@(2,3)_{\mathcal{B}} \;\wedge\; 0@(3,3)_{\mathcal{B}} \;\wedge\; \frac{0}{\widehat{Q_0}}@(4,3)_{\mathcal{B}}$
- $(x_2 = 0)@[2] \quad \equiv \quad \frac{\square}{\widehat{q_1}}@(4,3)_{\mathcal{B}} \;\wedge\; 0@(5,3)_{\mathcal{B}} \;\wedge\; \frac{0}{\widehat{Q_0}}@(6,3)_{\mathcal{B}}$
- $(x_2 = 0)@[3] \quad \equiv \quad 0$

 doi:10.20944/preprints202507.2367.v1

6.2.2. Boxed Definition

The sFHB Algorithm Applied to ϕ

TM $\widehat{N}$ tosses l coins, where l is the number of distinct propositional variables in ϕ. Consider the corresponding basic $(3l+1) \times (3l+3)$ tableau $\mathcal{B}$, along with the extended $(3 \cdot (3l+1)+1) \times (3l+3)$ tableau $\mathcal{T}$. Let the original FHB algorithm $\mathcal{A}$ run on $\widehat{N}$ and $\mathcal{T}$, while adhering to the correlated coin-tossing constraints encoded by $\psi^2_{extra}[\mathtt{sFHB}]$, in the following modified manner:

1. Guess a row index $\widehat{r}$ such that $\frac{\#}{\widehat{q_1}}@(\widehat{r},\widehat{c})_{\mathcal{B}}$ is fixed,
   - $l+1 \leq \widehat{r} \leq 3l+1$,
   - l is even iff $\widehat{r}$ is odd,

   where $\widehat{c}$ indexes the rightmost column. Restrict attention to rows $r \in \{1,\ldots,\widehat{r}\}$ of the *mini tableau* $\mathcal{B}$, denoted $\mathcal{B}_{\widehat{r}}$, until backtracking from the guessed $\widehat{r}$.
2. Restrict $\mathcal{A}$ to operate within $\mathcal{T}_{\widehat{r}}$, the extended tableau induced by $\mathcal{B}_{\widehat{r}}$. The algorithm proceeds columnwise, starting with the central column $\lceil \frac{\widehat{c}}{2} \rceil$. Subsequently, in stage two, the columns $\lceil \frac{\widehat{c}}{4} \rceil$ and $\lceil \frac{3\widehat{c}}{4} \rceil$ are visited. In stage three, the columns $\lceil \frac{\widehat{c}}{8} \rceil$, $\lceil \frac{3\widehat{c}}{8} \rceil$, $\lceil \frac{5\widehat{c}}{8} \rceil$, and $\lceil \frac{7\widehat{c}}{8} \rceil$ are visited. And so on.
3. For each column c visited, and for a selected row r in $\mathcal{B}_{\widehat{r}}$, guess between the first arrangement ((42)) and the second arrangement ((43)), and fill in $cell[c,r]$, $cell[c,r+1]$, and $cell[c,r+2]$ accordingly. Furthermore, populate all cells above $cell[r,c]$ with the blank symbol. Then, depending on the selected arrangement, fill all cells below $cell[r+2,c]$ with the symbol 1 (respectively, 0) for the first (respectively, second) arrangement.
4. Invoke the solver $\mathcal{H}$ to check the satisfiability of the formula $\psi_{trim} \wedge \ldots$, where the ellipsis represents all guesses under consideration. Specifically, satisfying ψ_{trim} entails satisfying $\psi^2_{extra}[\mathtt{sFHB}]$.
5. If backtracking occurs, all symbols assigned to column c of $\mathcal{T}_{\widehat{r}}$, as well as those placed elsewhere in $\mathcal{T}_{\widehat{r}}$, are rolled back accordingly.

6.2.3. Elaborating Step 4

Recall the following example of a correlated coin-tossing constraint. We have transformed the non-Horn formula (45) into the Horn formula (46).

$$x_2 \wedge \overline{x_1} \rightarrow x_3 \tag{45}$$

$$\bigwedge_{1 \leq \beta_2 \leq l} \bigwedge_{1 \leq \beta_1 \leq l} [\,(x_2 = 1)@[\beta_2] \;\wedge\; (x_1 = 0)@[\beta_1] \;\rightarrow\; 1@(4,2)_{\mathcal{B}}\,]. \tag{46}$$

Suppose the sFHB algorithm guesses $(x_2 = 1)@[1]$ and $(x_1 = 0)@[1]$. Conceptually, the solver $\mathcal{H}$ then injects the symbol 1 into $cell[4,2]$ of the basic tableau $\mathcal{B}$.

6.2.4. Complexity

Regarding the computational complexity of the sFHB algorithm, a quick, conservative estimation of the combinatorial cost associated with the separation of concerns (Section 6.2.1) is given by:

$$T(l^2) = 2 \cdot O(l) \cdot \left[T\left(\frac{l^2}{2}\right) + T\left(\frac{l^2}{2}\right) \right], \tag{47}$$

where the constant 2 refers to the two arrangements (42) and (43), and the plus sign (rather than a multiplication sign) embodies the separation of concerns.

Upon a change of variables, the recurrence relation transforms into:

$$T(p) = O(\sqrt{p}) \cdot \left[T\left(\frac{p}{2}\right) \right], \tag{48}$$

 doi:10.20944/preprints202507.2367.v1

which, by the standard derivation in Appendix F, yields the following approximate solution:

$$T(p) \approx C^{(\log p)^2} = p^{O(\log p)}, \tag{49}$$

for some constant $C > 0$. The result is superpolynomial but subexponential.

This already establishes that, at a minimum, an exponential-time deterministic TM exists that outperforms every $\langle N, k \rangle$ machine. Notably, this conclusion is reached independently of the cost analysis presented in [1]. Furthermore, Lemma 2 enables us to strengthen this result even further.

Lemma 2. *The runtime of the* `sFHB` *algorithm is bounded above by that of* `rFHB`.

Proof. It is sufficient to demonstrate that, asymptotically, the size of $\psi^2_{extra}[\texttt{sFHB}]$ does not exceed that of $\psi^2_{extra}[\texttt{rFHB}]$. This claim follows directly from the cost analyses presented in Sections 5.2 and 5.4. □

Hence, the tight upper bound for `rFHB` stated in inequality (38) of Theorem 3 also applies to the `sFHB` algorithm, which is conceptually much simpler to teach and implement.

7. Closing Remarks

A polynomially bounded tableau inherently lacks the capacity to concretely represent an exponential number of computation paths. As a result, the coin-tossing behavior of any $\langle N, k \rangle$ machine must exhibit significant correlation. Crucially, as the length of the input w to machine N increases, the portion of the tableau attributable to coin tosses becomes increasingly negligible in comparison to its total size.

In this article, we have demonstrated how to quantify these properties and derive theoretical insights accordingly. To ensure a level of rigor absent from our previous work [1], we have presented our analysis in great detail. A short summary of our findings will be developed in follow-up work.

Funding: This research received no funding.

Acknowledgments: Forthcoming.

Conflicts of Interest: The author declares no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

TM	Turing Machine
NP	Nondeterministic Polynomial

Appendix A.

Definition A1. *Cf. [10] (p. 259, 271). Variables that can take on the values* `TRUE` *and* `FALSE` *are called* **Boolean variables**. *We represent* `TRUE` *by* 1 *and* `FALSE` *by* 0. *The* **Boolean operations** `AND`, `OR`, *and* `NOT`, *represented by the symbols* $\wedge$, $\vee$, *and* $\neg$, *respectively, are described in the standard manner. We use the overbar as a shorthand for the* $\neg$ *symbol, so* $\overline{x}$ *means* $\neg x$. *A* **Boolean formula** *is an expression involving Boolean variables and operations. It is* **satisfiable** *if some assignment of* 0s *and* 1s *to the variables makes the formula evaluate to* 1. *We say the assignment* satisfies ϕ. *The* **satisfiability problem** *is to test whether a Boolean formula is satisfiable. Let* $\texttt{SAT} = \{\langle \phi \rangle \mid \phi \text{ is a satisfiable Boolean formula}\}$, *where* $\langle \phi \rangle$ *refers to a standard encoding of* ϕ.

Definition A2. *Cf. [10] (p. 273). A* **literal** *is a Boolean variable or a negated Boolean variable, as in* x *or* $\overline{x}$. *The former is called a* **positive literal**, *while the latter is called a* **negative literal**. *A* **clause** *is several literals connected with* $\vee$s, *as in* $(x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4)$. *A Boolean formula is in* **conjunctive normal form**, *called a* **cnf formula**, *if it comprises clauses with* $\wedge$s, *as in* $(x_1 \vee \overline{x_2} \vee \overline{x_3} \vee x_4) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6})$. *The*

 doi:10.20944/preprints202507.2367.v1

Boolean formula is a **3cnf formula** *if each clause has three literals, as in* $(x_1 \vee \overline{x_2} \vee \overline{x_3}) \wedge (x_3 \vee \overline{x_5} \vee x_6) \wedge (x_3 \vee \overline{x_6} \vee x_4)$. *A* **2cnf formula** *is an* AND *of clauses, where each clause is an* OR *of at most two literals.*

Definition A3. *Cf. [20] (p. 34–35). A* **(propositional) Horn formula** *is a cnf formula where every disjunction contains at most one positive literal.*
Also, $\texttt{HORNSAT} = \{\langle \phi \rangle \mid \phi \text{ is a satisfiable Horn formula}\}$.

Remark A1. *Horn clauses can be written as implications by the following equivalence (≡):*

$$\overline{x_1} \vee \ldots \vee \overline{x_k} \vee x \equiv (x_1 \wedge \ldots \wedge x_k) \rightarrow x$$

Theorem A1. *Cf. [20] (p. 35).* $\texttt{HORNSAT} \in \mathcal{P}$.

Definition A4. *Cf. Sipser [10] (p. 140). A* **deterministic Turing machine** *is an 8-tuple* $(Q, \Gamma, \Phi, \delta, T, q_0, q_{accept}, q_{reject})$, *with* Q, Γ, Φ, T *finite sets:*

> Q *is the set of states, and* Γ *is the input alphabet not containing the* blank symbol $\square$.
> Φ *is the tape alphabet, where* $\square \in \Phi$ *and* $\Gamma \subsetneq \Phi$.
> $\delta : Q \times \Phi \rightarrow Q \times \Phi \times \{+, -\}$ *is the transition function.*
> *Every transition in* δ *is accompanied by a distinct label* t.
> T *is the* label set, *containing all such labels.*
> $q_0 \in Q$ *is the start state.*
> $q_{accept} \in Q$ *is the accept state.* $q_{reject} \in Q$ *is the reject state, with* $q_{reject} \neq q_{accept}$.

As a Turing machine computes, changes occur in the current state, the current tape contents, and the current head location. A setting of these three items is called a **configuration** *of the Turing machine. The tape of the Turing machine is one-way infinite, from left to right. Specifically, for each input* $w_0 w_1 \ldots w_{n-1}$ *of length* n, *machine* M *starts in configuration* $\frac{\square}{q_0}\square\square\ldots\square$, *for* $n = 0$, *and in configuration* $\frac{w_0}{q_0} w_1 \ldots w_{n-1}\square\square\ldots\square$, *for* $n > 0$. *In both cases, the notation* $\frac{s}{q_0}$, *with* $s \in \Phi$, *signifies that the head is located at the tape cell containing symbol* s, *while the machine resides in state* q_0. *Machine* M *neither starts in* q_{accept} *or* q_{reject}, *nor progresses beyond either one of these states. Specifically, once* M *reaches* q_{accept}, *it remains active solely in that state. Likewise for* q_{reject}. *We take* q_{accept} *to be some* q_m *with* $m > 0$ *and similarly for* q_{reject}. *Input word* w *is considered accepted when* M *on* w *reaches* q_{accept}. *We write* $L(M)$ *to denote the language accepted by* M. *We use notation* $t : (q_1, x) \rightarrow (q_2, y, \mu)$ *when referring to some transition in* δ *with label* $t \in T$ *and movement* $\mu \in \{+, -\}$. *The plus sign (minus sign) signifies a movement to the right (to the left).*

Definition A5. *Cf. Sipser [10] (p. 150). A* ***nondeterministic Turing machine*** *is an 8-tuple,* $(Q, \Gamma, \Phi, \delta, T, q_0, q_{accept}, q_{reject})$. *At any point in a computation, the machine may proceed according to several possibilities. The transition function for the machine has the form* $\delta : Q \times \Phi \rightarrow \mathcal{P}(Q \times \Phi \times \{+, -\})$, *where* $\mathcal{P}(\,)$ *denotes the power set. The computation of the machine is a tree whose branches correspond to different possibilities for the machine. If some branch of the computation leads to the state* q_{accept}, *the machine accepts its input.*

In conformity with Definition A4, each transition in set δ *is accompanied by a distinct label* t, *and* T *is now called the* ***general label set****, containing all such labels. For instance, consider notation* $t : (q, x) \rightarrow \{(q_1, y_1, \mu_1), (q_2, y_2, \mu_2)\}$, *with label* $t \in T$, *states* $q, q_1, q_2 \in Q$, *symbols* $x, y_1, y_2 \in \Phi$, *movements* $\mu_1, \mu_2 \in \{+, -\}$, *and with tuple* (q_1, y_1, μ_1) *different from* (q_2, y_2, μ_2). *This notation captures the* ***nondeterministic transition*** *encompassing the deterministic transitions* $t[1] : (q, x) \rightarrow (q_1, y_1, \mu_1)$ *and* $t[2] : (q, x) \rightarrow (q_2, y_2, \mu_2)$. *We define the* ***basic label set,*** *denoted as* $T[\,]$, *as the set that encompasses the labels of all deterministic transitions, such as* $t[1]$ *and* $t[2]$.

Remark A2. *In our discussion, we employ the term "nondeterministic transition" explicitly, while abbreviating "deterministic transition" and "basic label set" to simply "transition" and "label set." We frequently omit*

 doi:10.20944/preprints202507.2367.v1

brackets for ease of reading; e.g., we write t and t' instead of $t[1]$ and $t[2]$. The crux is that each basic label is unique.

Definition A6. *Consider an arbitrary nondeterministic Turing machine, N, and its basic label set, $T[\,]$. For any label t in $T[\,]$ with corresponding signature $t : (q_{source}, s_{read}) \to (q_{target}, s_{write}, +)$ or $t : (q_{source}, s_{read}) \to (q_{target}, s_{write}, -)$, we let "$N$-source$(t)$," "$N$-target$(t)$," and "$N$-write$(t)$" stand for the symbols $\frac{s_{read}}{q_{source}}$, q_{target}, and s_{write}, respectively. When machine N is clear from the context, we shall simply note down "source(t)," "target(t)," and "write(t)," respectively.*

Definition A7. *Consider an arbitrary nondeterministic Turing machine, N, and its general label set, T. We let $T_{\mathtt{det}}$ denote the subset of T containing the labels of all deterministic instructions of N. We let $T^{+}_{\mathtt{det}}$, respectively $T^{-}_{\mathtt{det}}$, denote the subset of $T_{\mathtt{det}}$ containg the labels of all deterministic instructions of N whose movement is to the right $(+)$, respectively to the left $(-)$.*

Definition A8. *Let N be a nondeterministic Turing machine decider. Its* **running time** *is function $t : \mathcal{N} \to \mathcal{N}$, with $t(n)$ the maximum number of steps N uses on any branch of its computation, on any input of length n, before halting.*

Definition A9. *Cf. [10](pp. 251, 258). Let $t : \mathcal{N} \to \mathcal{R}^+$ be a function, with $\mathcal{R}^+$ denoting the set of nonnegative real numbers. Define the* **time complexity class**, $\mathtt{TIME}(t(n))$, *to be the collection of all languages that are decidable by an $O(t(n))$ time Turing machine. $\mathcal{P}$ is the class of languages that is decidable in polynomial time on a deterministic single-tape Turing machine. In other words, $\mathcal{P} = \bigcup_k \mathtt{TIME}\left(n^k\right)$.*

Theorem A2. *Cf. [20](p. 35).* $\mathtt{HORNSAT} \in \mathcal{P}$.

Definition A10. *Cf. [10](pp .265–267). A* **verifier** *for a language A is an algorithm V, where $A = \{w \mid V \text{ accepts } \langle w, c\rangle \text{ for some string } c\}$. We measure the time of a verifier only in terms of the length of w, so a* **polynomial time verifier** *runs in polynomial time in the length of w. A language A is* **polynomial verifiable** *if it has a polynomial time verifier. $\mathcal{NP}$ is the class of languages that have polynomial time verifiers.*

The remaining three items come from [10, p.266-276].

Theorem A3. *A language is in $\mathcal{NP}$ iff it is decided by some nondeterministic polynomial time Turing machine.*

Definition A11. $\mathtt{NTIME}(t(n)) = \{L \mid L \text{ is decided by a } O(t(n)) \text{ time nondeterministic TM}\}$. *A function $f : \Sigma^* \to \Sigma^*$ is a* **polynomial time computable function** *if some deterministic polynomial time Turing machine M exists that halts with just $f(w)$ on its tape, when started on any input w. A language B is* ***$\mathcal{NP}$-complete*** *if it satisfies two conditions: (1) B is in $\mathcal{NP}$, and (2) every A in $\mathcal{NP}$ is polynomial time reducible to B.*

Theorem A4. *If B is $\mathcal{NP}$-complete and $B \in \mathcal{P}$, then $\mathcal{P} = \mathcal{NP}$.*

Appendix B.

Appendix B.1. The Single Part

We define ψ^{single}_{extra} as follows:

$$\psi^{single}_{extra} \quad = \quad \frac{s}{q}@(3l-2, j) \;\to\; \bigwedge_{s'} \bigwedge_{q'} \bigwedge_{j' \neq j} \neg \frac{s'}{q'}@(3l-2, j'),$$

 doi:10.20944/preprints202507.2367.v1

where $s' \in \Phi$, $q' \in Q$, and the column index j' ranges from 1 to $n^k + 2$. This condition ensures that if the tape-state symbol $\frac{s}{q}$ is stored in $cell[3l-2, j]$, it is the only tape-state symbol in row $3l-2$. In other words, no tape-state symbol $\frac{s'}{q'}$ can be stored in any other cell within the same row.

Appendix B.2. The Left Part

We define ψ^{left}_{extra} as follows:

$$\psi^{left}_{extra} \;=\; \psi^{left}_{1} \;\wedge\; \psi^{left}_{2} \;\wedge\; \psi^{left}_{3}$$

On the one hand, we reason from row i both towards earlier rows ($-\delta$) and towards later rows ($+\delta$):

$$\psi^{left}_{1} \;=\; \bigwedge_{0<m<j} \left[\frac{s}{q}@(3l-2, j) \;\wedge\; s'@(3l-2, j-m) \;\rightarrow\; \bigwedge_{r(\delta)} s'@\left((3l-2) \underset{+}{-} 3\delta, j-m \right) \right],$$

where the restriction on δ, denoted by $r(\delta)$, is defined by the following inequalities:

- $0 \leq \delta \leq m-1$.
- $1 \leq (3l-2) - 3\delta \leq (3l-2) + 3\delta \leq 3n^k + 1$.

Additionally, the expression

$$s@\left(i \underset{+}{-} 3\delta, j \right)$$

is shorthand for:

$$\begin{aligned} s@(i - 3\cdot\delta_{min}(m), j) \;\wedge\; s@(i - 3\cdot\delta_{min}(m) + 1, j) \;\wedge\; s@(i - 3\cdot\delta_{min}(m) + 2, j) \;\wedge \\ \ldots \;\wedge\; s@(i + 3\cdot\delta_{max}(m) - 1, j) \;\wedge\; s@(i + 3\cdot\delta_{max}(m), j), \end{aligned}$$

where $\delta_{min}(m)$ is defined as the maximum (not the minimum) of the set:

$$\{\; \delta \mid 0 \leq \delta \leq m-1 \text{ and } 1 \leq i - 3\delta \leq 3n^k + 1 \;\}.$$

Similarly, $\delta_{max}(m)$ is defined as the maximum of the set:

$$\{\; \delta \mid 0 \leq \delta \leq m-1 \text{ and } 1 \leq i + 3\delta \leq 3n^k + 1 \;\}.$$

Example A1. *For instance, if $\frac{a}{q_5}@(i, j)$ and $s^*@(i, j-3)$ hold, with $i = 3l-2$, then the following must also hold:*

$$\bigwedge_{r(\delta)} [\, s^*@(i \underset{+}{-} 3\delta, j-3) \,],$$

where $r(\delta)$ is defined by the following inequalities:

- $0 \leq \delta \leq 2$.
- $1 \leq i - 3\delta \leq i + 3\delta \leq 3n^k + 1$.

The reader can verify that all relevant cells range from $cell[i - 3\cdot 2, j-3]$ at the top to $cell[i + 3\cdot 2, j-3]$ at the bottom. These $3\cdot 2 + 1 + 3\cdot 2 = 13$ cells correspond to the 13 crossed-out entries in column $j-3$ in Table 7.

 doi:10.20944/preprints202507.2367.v1

On the other hand, we reason from the earliest row ($i - 3 \cdot \delta_{min}(m)$) and, respectively, the latest row ($i + 3 \cdot \delta_{max}(m)$) toward row i, with $i = 3l - 2$:

$$\psi_2^{left} = \bigwedge_{0<m<j} \left[\frac{s}{q}@(3l-2,j) \ \wedge \ s'@((3l-2) - 3 \cdot \delta_{min}(m), j-m) \ \rightarrow \ s'@(3l-2, j-m) \right],$$

$$\psi_3^{left} = \bigwedge_{0<m<j} \left[\frac{s}{q}@(3l-2,j) \ \wedge \ s'@((3l-2) + 3 \cdot \delta_{max}(m), j-m) \ \rightarrow \ s'@(3l-2, j-m) \right].$$

The satisfiability of ψ_3^{left} ensures that if a specific cell, such as

$$cell[(3l-2) + 3 \cdot \delta_{max}(m), j-m],$$

for some fixed m, is filled with the tape symbol s^*, then—through a chain reaction partially facilitated by ψ_1^{left}—all crosses in column $j - m$ are replaced with the symbol s^*. A similar observation applies to the satisfiability of ψ_2^{left}, which affects the propagation from an earlier cell, i.e.,

$$cell[(3l-2) - 3 \cdot \delta_{min}(m), j-m].$$

Appendix B.3. The Right Part

Given the inherent symmetry of the problem, the formal definition of ψ_{extra}^{right} closely mirrors that of ψ_{extra}^{left} and is therefore omitted from this paper.

Appendix B.4. The Extend Part

$$\psi_{extra}^{extend} \ = \ \psi_{extra}^{-} \ \wedge \ \psi_{extra}^{-,-} \ \wedge \ \psi_{extra}^{+} \ \wedge \ \psi_{extra}^{+,+},$$

where—for $s, s' \in \Phi$ and $q, q' \in Q$—we have:

$$\psi_{extra}^{-} \ = \ \bigwedge_{s} \bigwedge_{q} \bigwedge_{s'} \bigwedge_{q'} \left[\frac{s}{q}@(3l-2,j) \ \wedge \ \frac{s'}{q'}@(3l+1,j-1) \ \rightarrow \ t^-(s,q)@(3l,j) \right],$$

$$\psi_{extra}^{-,-} \ = \ \bigwedge_{s} \bigwedge_{q} \bigwedge_{s'} \left[\frac{s}{q}@(3l-2,j) \ \wedge \ s'@(3l+1,j+1) \ \rightarrow \ t^-(s,q)@(3l,j) \right],$$

$$\psi_{extra}^{+} \ = \ \bigwedge_{s} \bigwedge_{q} \bigwedge_{s'} \bigwedge_{q'} \left[\frac{s}{q}@(3l-2,j) \ \wedge \ \frac{s'}{q'}@(3l+1,j+1) \ \rightarrow \ t^+(s,q)@(3l,j) \right],$$

$$\psi_{extra}^{+,+} \ = \ \bigwedge_{s} \bigwedge_{q} \bigwedge_{s'} \left[\frac{s}{q}@(3l-2,j) \ \wedge \ s'@(3l+1,j-1) \ \rightarrow \ t^+(s,q)@(3l,j) \right].$$

Here, $t^-(s,q)$ and $t^+(s,q)$ represent two distinct labels of the machine N, such that:

$$N\text{-source}\big(t^-(s,q)\big) \ = \ \frac{s}{q} \ = \ N\text{-source}\big(t^+(s,q)\big).$$

Recall Definition A6. The minus (plus) sign indicates that N moves to the left (right).

Appendix B.5. Consolidation

In summary, ψ_{extra}^1 is a Horn formula, with a size of $O(n^{4k})$, where the constant k corresponds to the running time n^k of machine N.

 doi:10.20944/preprints202507.2367.v1

Appendix C.

We can readily formulate several more $\overleftrightarrow{\downarrow}$ constraints. For instance, let us continue with $l = 3$ and Table 11. Suppose that, in the *mini tableau*, the coin toss outcome $x_2 = 1$ has materialized:

$$\overset{\square}{\widehat{q_1}}@(3,4)_{\mathcal{B}} \;\vee\; \overset{\square}{\widehat{q_1}}@(5,4)_{\mathcal{B}}.$$

Now, consider the cell c_{**}—positioned at row i_{**} and column j_{**}, beyond the confines of the first $\widehat{r}$ rows of the *mini tableau*—for which the following proposition holds:

$$\overset{b}{\overleftrightarrow{U_0}}@(i_{**}-1,\, j_{**}+1)_{\mathcal{B}}, \;\text{ with }\; b \in \{0,1\}.$$

In other words, relative to the cell c_{**}, one time step earlier and one cell to the right, the head is scanning a bit b while in state $\overleftrightarrow{U_0}$. This state of affairs is exemplified by the following configuration and $b = 0$:

$$\&\,\$\,cl_m\,\$\ldots\$\,cl_2\,\$\,a\,1\,\overset{0}{\overleftrightarrow{U_0}}\;\vee\;0\,1\;\vee\;0\;\vee\;\$\#\ldots\#.$$

In conformity with instructions $\overleftrightarrow{t_{8,0}}$ and $\overleftrightarrow{t_{8,1}}$—recall from Section 4.2.4:

$$\overleftrightarrow{t_{8,x}} : \left(\overleftrightarrow{U_0}, x\right) \to \left(\overleftrightarrow{U_0}, x, -\right), \;\; x \in \{\vee, \$, 0, 1\}$$

—we then posit that

$$\overset{1}{\overleftrightarrow{U_0}}@(i_{**},\, j_{**})_{\mathcal{B}}$$

must hold. Here, the tape-state symbol $\overset{1}{\overleftrightarrow{U_0}}$ signifies that the TM's head is scanning—from right to left—in search of the first occurrence of a symbol $\sigma \in \{a, \overline{a}\}$ located to the left of the bit 1 in cell c_{**}. (When such a σ is found, the machine overwrites it: replacing a with 0 and $\overline{a}$ with 1.)

Formally, these long-range dependencies are captured by the following $\overleftrightarrow{\downarrow}$ constraints—each a Horn formula with three literals:

$$\begin{aligned}
&\overset{\square}{\widehat{q_1}}@(3,4)_{\mathcal{B}} \;\wedge\; \overset{1}{\overleftrightarrow{U_0}}@(i_{**}-1,\, j_{**}+1)_{\mathcal{B}} \;\to\; \overset{1}{\overleftrightarrow{U_0}}@(i_{**},\, j_{**})_{\mathcal{B}},\\
&\overset{\square}{\widehat{q_1}}@(5,4)_{\mathcal{B}} \;\wedge\; \overset{1}{\overleftrightarrow{U_0}}@(i_{**}-1,\, j_{**}+1)_{\mathcal{B}} \;\to\; \overset{1}{\overleftrightarrow{U_0}}@(i_{**},\, j_{**})_{\mathcal{B}},\\
&\overset{\square}{\widehat{q_1}}@(3,4)_{\mathcal{B}} \;\wedge\; \overset{0}{\overleftrightarrow{U_0}}@(i_{**}-1,\, j_{**}+1)_{\mathcal{B}} \;\to\; \overset{1}{\overleftrightarrow{U_0}}@(i_{**},\, j_{**})_{\mathcal{B}},\\
&\overset{\square}{\widehat{q_1}}@(5,4)_{\mathcal{B}} \;\wedge\; \overset{0}{\overleftrightarrow{U_0}}@(i_{**}-1,\, j_{**}+1)_{\mathcal{B}} \;\to\; \overset{1}{\overleftrightarrow{U_0}}@(i_{**},\, j_{**})_{\mathcal{B}}.
\end{aligned}$$

Conservatively, this yields a complexity of:

$$O(n^2) \times l \times l \times 2 \;\;=\;\; O(n^4).$$

Appendix D.

To solve the recurrence relation, we begin with the following setup:

$$T(p) = \kappa_0 \cdot \sqrt{p} \cdot T\left(\frac{p}{2(\log l)^2}\right)^2, \quad \text{with} \quad T(1) = O(1),$$

where κ_0 is a constant.

 doi:10.20944/preprints202507.2367.v1

Step 1: Change of Variables

Let us set:

$$q = \log p \quad \Rightarrow \quad p = 2^q, \quad \text{so} \quad \sqrt{p} = 2^{q/2}.$$

Now consider the recurrence:

$$\frac{p}{2(\log l)^2} = \frac{2^q}{2(\log l)^2} = 2^{q - \log(2(\log l)^2)} = 2^{q - \log 2 - 2\log\log l}.$$

Let:

$$c = \log 2 + 2\log\log l = 1 + 2\log\log l.$$

Then the recurrence becomes:

$$T(2^q) = \kappa_0 \cdot 2^{q/2} \cdot T(2^{q-c})^2.$$

Taking logarithms (base 2) on both sides:

$$\log T(2^q) = \log \kappa_0 + \frac{q}{2} + 2\log T(2^{q-c}).$$

Let:

$$S(q) = \log T(2^q),$$

so the recurrence becomes:

$$S(q) = \log \kappa_0 + \frac{q}{2} + 2S(q - c).$$

Step 2: Solve the Linear Recurrence

Unrolling the recurrence for n steps:

$$S(q) = \sum_{i=0}^{n-1} 2^i \left(\log \kappa_0 + \frac{q - ic}{2} \right) + 2^n S(q - nc).$$

To reach the base case $S(q - nc) = O(1)$, choose:

$$n \geq \frac{q}{c} \quad \Rightarrow \quad n = \left\lceil \frac{q}{c} \right\rceil.$$

Then:

$$S(q) = O(2^n q) = O\left(2^{q/c} q\right).$$

Recalling that $q = \log p$, we obtain:

$$S(q) = \log T(p) = O\left(p^{1/c} \log p\right).$$

Exponentiating both sides:

$$T(p) = 2^{S(\log p)} = \exp_2\left(O\left(p^{1/c} \log p\right)\right),$$

where:

$$c = 1 + 2\log\log l.$$

Thus, the solution is:

 doi:10.20944/preprints202507.2367.v1

$$\boxed{T(p) = \exp_2\Big(O\Big(p^{1/(1+2\log\log l)} \cdot \log p\Big)\Big)}.$$

Asymptotic Behavior as $l \to \infty$

As $l \to \infty$, we have:

$$\log l \to \infty \quad \Rightarrow \quad \log\log l \to \infty,$$

so:

$$\frac{1}{1+2\log\log l} \to 0.$$

Then:

$$p^{1/(1+2\log\log l)} \to 1,$$

and thus:

$$T(p) = \exp_2(O(\log p)) = O(p^{\alpha}),$$

for some constant $\alpha > 0$.

Hence, asymptotically:

$$\boxed{T(p) = p^{O(1)}}$$

as $l \to \infty$. This implies that $T(p)$ is polynomial in p, and the degree becomes smaller as l increases.

Appendix E.

We apply Theorem 2 iteratively to establish the validity of Theorem 3.

Appendix E.1. First Iteration

Theorem 2 conveys that any `3-SAT` solver $\langle N_1, 1\rangle$, with state set Q_1 and tape alphabet Φ_1—operating with $O(n)$ nondeterministic binary guesses followed by $O(n)$ deterministic verification time—can be simulated deterministically by some M_1 via the `rFHB` algorithm.

Via a single application of Theorem 2, the resulting deterministic TM M_1 runs in at most

$$K_1^{n^{0.67}} = 2^{(\log K_1)\cdot n^{0.67}} = 2^{\alpha_1 \cdot n^{0.67}}$$

time, where:

- The constant K_1 depends on the sizes of Q_1 and Φ_1.
- The constant $\alpha_1 = \log K_1 > 0$.

Inspired by the existence of deterministic machine M_1, we can now construct a nondeterministic polynomial time TM N_2, with state set Q_2 and tape alphabet Φ_2, where

$$|Q_1| \leq |Q_2| \quad \text{and} \quad \Phi_1 \subseteq \Phi_2.$$

This machine N_2 is functionally equivalent to M_1, and hence is also a `3-SAT` solver.

At first glance, N_2 operates with

$$O\Big(\alpha_1 \cdot n^{0.67}\Big) \;=\; O\Big(n^{0.67}\Big)$$

nondeterministic binary guesses, followed by

$$\nu = O(n^{\kappa_1})$$

deterministic verification time. Recall Claim 3.

 doi:10.20944/preprints202507.2367.v1

Upon closer inspection, N_2 operates with, say,

$$O\left(n^{0.70}\right)$$

nondeterministic binary guesses. An arbitrary small but nonzero increase in the number of guesses (from 0.67 to 0.70) is required here, for the following reason: N_2 guesses (and correctly so, in the satisfiable case) the guesses made by `rFHB` relative to N_1. Although the total number of holes is $O(n^{0.67})$, rather than linear in n, an additional $O(\log l)$ bits are needed to uniquely identify each guess within the tableau. Consequently, N_2 makes

$$O(\log l) \times O(n^{0.67})$$

binary guesses, which we simplify to $O(n^{0.70})$ for subsequent analysis.

To recatipulate, functionally equivalent machines N_1 and N_2 require $O(n)$ and $O(n^{0.70})$ guesses, respectively. For sufficiently large n, machine N_2 makes fewer guesses than N_1.

Appendix E.2. Second Iteration

Via a second application of Theorem 2, we may infer the existence of a deterministic machine M_2 that simulates the nondeterministic machine N_2. This machine M_2 runs in at most

$$2^{\alpha_2 \cdot (n^{0.70})^{0.67}} \times \nu \;=\; 2^{\alpha_2 \cdot n^{0.47}} \times \nu$$

time, where:

- The constant $\alpha_2 = \log K_2 > 0$.
- K_2 depends on the sizes of Q_2 and Φ_2, with $K_1 < K_2$.

Inspired by the existence of deterministic machine M_2, we can now construct a functionally equivalent nondeterministic polynomial time TM N_3, with state set Q_3 and alphabet Φ_3, where

$$|Q_2| \leq |Q_3| \quad \text{and} \quad \Phi_2 \subseteq \Phi_3.$$

This machine N_3 operates, not with

$$O\left(n^{0.47}\right),$$

but with, say,

$$O\left(n^{0.50}\right),$$

nondeterministic binary guesses, followed by

$$\nu + \nu \;=\; 2\nu$$

verification time. Again, the rationale for the slight increase in the exponent is due to the extra bits that are needed to uniquely identify each guess within the tableau.

To recatipulate, N_2 and N_3 require $O(n^{0.70})$ and $O(n^{0.50})$ guesses, respectively. For sufficiently large n, machine N_3 makes fewer guesses than N_2.

Appendix E.3. Third Iteration

Via a third application of Theorem 2, we may infer the existence of a deterministic TM M_3 that simulates the nondeterministic machine N_3. And so on.

 doi:10.20944/preprints202507.2367.v1

Appendix E.4. The j-th Iteration

Inspired by the existence of deterministic machine M_j, we can now construct a functionally equivalent, nondeterministic polynomial time TM N_{j+1}, with state set Q_{j+1} and tape alphabet Φ_{j+1}, where

$$|Q_j| \leq |Q_{j+1}| \quad \text{and} \quad \Phi_j \subseteq \Phi_{j+1}.$$

This machine N_{j+1} operates, not with

$$O\left(n^{(0.67)^j}\right),$$

but with, say,

$$O\left(n^{(0.70)^j}\right)$$

nondeterministic binary guesses, followed by

$$j \cdot \nu$$

verification time.

Appendix E.5. Consolidation

Take $j^* = \kappa \times (\log n)$, for some sufficiently large constant κ. Then machine N_{j^*+1} performs $O(1)$ guesses, followed by

$$\kappa \times (\log n) \times \nu \quad = \quad O(n^{\kappa_1+1})$$

deterministic verification time. That is, N_{j^*+1} is a deterministic polynomial time TM.

Appendix F.

Solve the recurrence relation:

$$T(p) = \kappa \cdot \sqrt{p} \cdot T\left(\frac{p}{2}\right)$$

where κ is a constant.

Solution

Assume $p = 2^n$. Then the recurrence becomes:

$$T(2^n) = \kappa \cdot \sqrt{2^n} \cdot T(2^{n-1}) = \kappa \cdot 2^{n/2} \cdot T(2^{n-1})$$

Apply the recurrence repeatedly:

$$\begin{aligned} T(2^n) &= \kappa \cdot 2^{n/2} \cdot \kappa \cdot 2^{(n-1)/2} \cdot T(2^{n-2}) \\ &= \kappa^2 \cdot 2^{n/2+(n-1)/2} \cdot T(2^{n-2}) \\ &= \cdots \\ &= \kappa^n \cdot \prod_{i=0}^{n-1} 2^{(n-i)/2} \cdot T(1) \end{aligned}$$

Now simplify the exponent:

$$\sum_{i=0}^{n-1} \frac{n-i}{2} = \frac{1}{2} \sum_{j=1}^{n} j = \frac{1}{2} \cdot \frac{n(n+1)}{2} = \frac{n(n+1)}{4}$$

Therefore:

$$T(2^n) = \kappa^n \cdot 2^{n(n+1)/4} \cdot T(1)$$

 doi:10.20944/preprints202507.2367.v1

Substituting back $n = \log p$, we get:

$$T(p) = \kappa^{\log p} \cdot 2^{\frac{\log p(\log p+1)}{4}} \cdot T(1)$$

Asymptotic Form

As $p \to \infty$,

$$\boxed{T(p) = 2^{\Theta((\log p)^2)} = p^{O(\log p)},}$$

which implies that the growth rate is *quasi-polynomial*—faster than any polynomial, yet slower than exponential.

References

1. Daylight, E.G. Tableau with Holes: Clarifying NP-Completeness. *Symmetry* **2025**, *17*.
2. Daylight, E. Injecting Observers into Computational Complexity. *Philosophies* **2025**, *10*. https://doi.org/10.3390/philosophies10040076.
3. Dean, W. Computational Complexity Theory. *The Stanford Encyclopedia of Philosophy* **2016**.
4. Fortnow, L. Why Can't We Break Cryptography? https://blog.computationalcomplexity.org/2025/04/why-cant-we-break-crytptography.html, 2025. Accessed: 2025-07-01.
5. Li, M.; Vitanyi, P. *An Introduction to Kolmogorov Complexity and Its Applications*, 4 ed.; Texts in Computer Science, Springer, 2019. https://doi.org/10.1007/978-3-030-11298-1.
6. Daylight, E.; Koolen, W.; Vitányi, P. Time-Bounded Incompressibility of Compressible Strings and Sequences. *Information Processing Letters* **2009**, *109*, 1055–1059.
7. Cook, S. The Complexity of Theorem-Proving Procedures. In Proceedings of the Proc. 3rd Annual ACM Symposium on Theory of Computing. Association for Computing Machinery, 1971, pp. 151–158.
8. Levin, L. Universal Sorting Problems. *Problems of Information Transmission* **1973**, *9*, 265–266. English translation of original in *Problemy Peredachi Informatsii*.
9. Fortnow, L.; Homer, S. A Short History of Computational Complexity. *Bulletin of the European Association for Theoretical Computer Science* **2003**, *80*, 95–133.
10. Sipser, M. *Introduction to the Theory of Computation*; Thomson Course Technology, 2006.
11. Papadimitriou, C. *Computational Complexity*; Addison Wesley Longman, 1994.
12. Hopcroft, J.; Motwani, R.; Ullman, J. *Introduction to Automata Theory, Languages, and Computation*; Addison Wesley / Pearson Education, 2007.
13. Aaronson, S. *Quantum Computing since Democritus*; Cambridge University Press, 2013.
14. Dean, W. Algorithms and the Mathematical Foundations of Computer Science. In *Gödel's Disjunction*, First ed.; Horsten, L.; Welch, P., Eds.; Oxford University Press, 2016.
15. Tall, D. *How Humans Learn to Think Mathematically: Exploring the Three Worlds of Mathematics*; Cambridge University Press, 2013.
16. Turner, R. Computational Abstraction. *Entropy* **2021**, *23*, 213. https://doi.org/10.3390/E23020213.
17. Linnebo, Ø.; Shapiro, S. Actual and Potential Infinity. *Noûs* **2019**, *53*, 160–191.
18. Fortnow, L. Can you feel the machine? https://blog.computationalcomplexity.org/2024/03/can-you-feel-machine.html, 2024. Accessed: 2024-08-22.
19. Hill, R.K. The Imperativity of Algorithms. https://cacm.acm.org/blogcacm/the-imperativity-of-algorithms/, 2023. Accessed: 2024-08-22.
20. Grädel, E. *Complexity Theory: WS 2009/10*; Mathematische Grundlagen der Informatik, RWTH Aachen, creativecommons.org, 2009.
21. Dowling, W.; Gallier, J. Linear-Time Algorithms for Testing the Satisfiability of Propositional Horn Formulae. *J. Logic Programming* **1984**, *1*, 267–284.

 doi:10.20944/preprints202507.2367.v1

22. Daylight, E. Dijkstra's Rallying Cry for Generalization: the Advent of the Recursive Procedure, late 1950s – early 1960s. *The Computer Journal* **2011**, *54*, 1756–1772.
23. Shapiro, S. Acceptable notation. *Notre Dame Journal of Formal Logic* **1982**, *23*, 14–20. https://doi.org/10.1305/NDJFL/1093883561.